US ARMY
LABORATORY COMMAND

CONTRACT REPORT NUMBER 3-89
PREPARED FOR THE

HUMAN ENGINEERING LABORATORY

BY

COMMUNICATIONS DESIGN CENTER
CARNEGIE MELLON UNIVERSITY
PITTSBURGH, PA 15213

HELP DESIGN SOFTWARE PROJECT

30 NOVEMBER 1988

DAAA1-86-K-0019AC85

Approved for public release;
distribution is unlimited.

ABERDEEN PROVING GROUND, MARYLAND 21005-5001

Help Design Software Project

researched, designed and written by

**Thomas M. Duffy, James Palmer, Brad Mehlenbacher,
Guojun Zhang, Ann Aaron, Maria Truschel, Karen Denchfield**

**Communications Design Center
Carnegie Mellon University
Pittsburgh, PA 15213**

November 30, 1988

**Funded in part by the United States Army
Human Engineering Laboratory, Aberdeen Proving Grounds, MD
under contract # DAAA1-86-K-0019 AC85.
Contract monitor: Maureen Larkin.**

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS None.
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Carnegie-Mellon University 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION U.S. Army Human Engineering Laboratory
6c. ADDRESS (City, State, and ZIP Code) Communications Design Center Carnegie-Mellon University Pittsburgh, PA 15213	7b. ADDRESS (City, State, and ZIP Code) ATTN: SLCHE-CS Aberdeen Proving Ground, MD 21005-5001
8a. NAME OF FUNDING / SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER Contract # DAAA1-86-K-0019AC85
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)

Help Design Software Project (Unclassified)

12. PERSONAL AUTHOR(S) Thomas M. Duffy, James Palmer, Brad Mehlenbacher, Guojun Zhang, Ann Aaron, Maria Truschel, Karen Denchfield

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Interim	FROM 4 Apr 87 TO 30 Nov 88	1988, Nov, 30	270

16. SUPPLEMENTARY NOTATION

This task was performed under the contract title: On-Line Help for Interactive Systems

17. COSATI CODES		
FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)

On-Line Help, HELP Systems, human-computer interface, human factors

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Interactive computer systems were once used only by highly-trained specialists. Today, complex computer systems are used by people with varying levels of skill and experience. Adequate on-line help systems must be designed so new or infrequent users will be able to effectively use the systems without lengthy training or experimentation. The purpose of this project was to perform a systematic investigation of on-line help and to develop guidelines for the design of on-line help systems for interactive computer systems. Three topics were hypothesized and investigated--

1. The help database should be arranged so that it can be efficiently maintained without sacrificing the capacity for many kinds of access and it should appear consistently so that entries can be added or amended easily. Alternate organizational strategies that balanced user-friendliness with ease of maintenance were developed and tested.

(continued on reverse side)

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (Include Area Code) / 22c. OFFICE SYMBOL

19. (Continued from first page)

2. Each entry in the database should be structured for effective scanning and interpretation. Alternate presentation formats for on-line help information were evaluated. Optimal display strategies are reported.

3. The type of information in the database should be specific to the users for which the on-line help system is designed. Procedures for assessing the kinds of information that will be useful were developed. Access channels to the on-line help database were designed.

Table of Contents

Overview of Year 2 Activities

by

Thomas M. Duffy, James Palmer, Brad Mehlenbacher,
Ann Aaron, Maria Truschel, Karen Denchfield.

Communications Design Center
Carnegie Mellon University
Pittsburgh, PA 15213

1 March 1988

This work was funded in part by the United States Army
Human Engineering Laboratory, Aberdeen Proving Grounds, MD
under contract DAAA1-86-K-0019.
Contract monitor: Maureen Larkin.

Our contract, Online Help for Interactive Systems (DAAA1-86-k-0019) began March 7, 1986. This report is a summary of the second year of activity.

One of the major products planned for in this project was the development of a kit of help design tools that can be used to evaluate any online help program. This was to include the development of a prototype help system.

The first year of this contract focused primarily on defining and refining the problem domain. We initiated a major literature review, collected online help design guidelines, reviewed existing help systems, and interviewed professionals who had developed help systems. That work contributed substantially to the data base developed during the second year. Perhaps, more importantly, the work contributed to the development of a conceptual and theoretical framework on the definition of online help and the design and evaluation of help systems. This framework has been central to the work during this second year.

There were also very frustrating aspects of the first year of work. We were ready to begin building model help systems and a design aiding system, but we simply could not identify any tools to support the effort. We lost significant time trying to locate tools for building and testing prototypes. This lost time included three months of frustration in trying to use the most widely recommended rapid prototyper, Rapid-Use. At the end of year one we recommended scaling down this component of the contract because there were not any tools available to support that goal of the contract within the cost and time constraints.

The second year of effort to build the design and evaluation system has been as fruitful as the first year of effort on that task was frustrating. We have made tremendous strides in implementing the system much along the lines originally planned. Indeed, developing the systems, and documenting the conceptual underpinnings of the products has been the major focus during the second year.

The progress has been made possible in large part by the introduction of HyperCard by Apple Computer. HyperCard is an object oriented programming system, supported by a very natural programming language, for the development of relational databases. Because of the success of our work with HyperCard, we have begun to replace our Sun Microsystems with Apple Macintoshes.

The work during this year had included front end analysis to contribute to the refinement of the project goals, experimental evaluation of alternative design strategies, and the development of the design and evaluation tools that are the primary focus of the contract. The major focus during the last several months has been on the design and evaluation tools. We feel we have made substantial progress in developing the theoretical underpinnings of these tools and in the actual tool development. The system that we are developing has three basic components:

- *Help Design System* to assist developers in incorporating sound design principles in their online help.

- *Help Evaluation System* to assist developers and end users in diagnosing the strengths and weakness of any online help system.

- *Help Prototype Testing System* to assist in testing alternative design strategies for components of online help.

We have provided papers describing the conceptual basis for each of these systems, as well as prototypes of the actual systems. We also briefly describe each component later in this overview.

Deliverables

Deliverables included with this report or to be submitted before 1 May are as follows:

Reports

- The Design and Evaluation of Online Help for Unix EMACS: Capturing the User In Menu Design. James Palmer, Thomas M. Duffy, Kathleen Gomoll, Jessica Richards-Palmquist, John Trumble. (delivered)

- Developing the Help Design Software. Thomas M. Duffy, James Palmer, Brad Mehlenbacher, Guojun Zhang, Ann Aaron, Maria Truschel, Karen Denchfield. (delivered)

- Writing Online Information: Expert Strategies. Thomas M. Duffy, Tom Gomoll, Kathleen Gomoll, James Palmer, Ann Aaron. (delivered)

- Theory and Methodology of Evaluating Online Help. Thomas M. Duffy, James Palmer, John Trumble. (delivered)

- The Help Design Evaluation Survey (Versions A&B). Thomas M. Duffy. (delivered)

- Finding Information on a Menu: Linking Menu Organization to the User's Goal. Brad Mehlenbacher, Thomas M. Duffy, James Palmer. (available 1 May)

- An Analysis of Online Help Programs. Thomas M. Duffy, John Trumble, James Palmer. (available 1 May)

Software

- The Help Design System. Thomas M. Duffy, James Palmer, Brad Mehlenbacher, Ann Aaron, Maria Truschel, Karen Denchfield. (delivered)

- The Menu Tester (Experimental Version). Brad Mehlenbacher, Thomas M. Duffy, James Palmer. (delivered)

The Online Help Team

We have been lucky to have a very strong development team working on the project this year. The staff funded at least partially through the contract are:

- Tom Duffy (Principal Investigator)
- Jim Palmer (PhD student and Associate Principal Investigator)
- Brad Mehlenbacher (PhD student)
- Guojun Zhang (post doctoral student)
- Maria Truschel (Masters student)
- Karen Denchfield (Masters student)
- Ann Aaron (Masters student)

We have also had an enthusiastic response to a course offering on the Design of Online Help during the spring 1988 semester. Our team has been very effectively augmented by the following people enrolled in that course.

- Bill Hefley (project manager, Software Engineering Institute)
- Alan Houser (CMU staff, Computer Science)
- Debbi Smelson (senior)
- Shari Miller (senior)

Presentations

We have made or are scheduled to make presentations on the Help Design Software at the following places.

Software Engineering Institute, Carnegie Mellon University (colloquium presented 22 February 1988)

- CHI 88, The Computer User Interaction Annual Conference, Washington, D.C., 15-18 May 1988. (We will be presenting a poster session, demonstrating the software)

- Massachusetts Institute of Technology, Conference on Writing Computer Documentation, 15 August 1988. (We will be presenting two 2 hour workshops demonstrating the software).

Online Help Defined

Online help is a very broad term that has been applied to label many different aspects of the computer system. One of the first requirements in this project was to define the term and thereby specify the domain of our efforts. We define online help as a program and database that is a resource for using an application program or some piece of hardware (Duffy and Langston, 1985). That is, it is a program that, at a minimum, permits access to and navigation through a database. The program and database are functionally distinct from the application software and access to the database is initiated by the user.

Online help is distinguished from error correction or error feedback in that error correction is: a.) typically a part of the application program, rather than a separate program; and b.) system, rather than user, activated. Thus, online help is a computer based information system accessed by a soldier and designed to provide the soldier with assistance in completing a task within another computer program.

The goal for online documentation is job aiding; it is a source of reference information and procedural instructions. We distinguish online help from tutorials or training on the basis of this job aiding vs instructional function. The user of online help is seen as a soldier in the middle of a job rather than a soldier learning about a job.

We do not distinguish between online documentation and online help. We will classify a system as online help as long as it is a functionally distinct program and database and the goal in the design is to aid job performance. If a technical manual that was designed to aid a soldier in using a computer system were placed online, we would classify that online manual as an online help system. We would hasten to add, however, that we would expect such an online help system to be ineffective: the design of an online help system is quite different from the design of a technical manual (Duffy, Gomoll, Gomoll, Palmer, Aaron, 1988).

With this definition of online help in mind, we now review the status of our work in developing the help design and evaluation systems.

Help Design and Evaluation Tools

The Help Design System

The Help Design System is being developed as a job aid to be used in the software engineering environment for planning and designing online help programs. The goal of the system is to both present sound guidance on the design of online help and to help developers incorporate that guidance in their product. The intended audience for the tool is writers, programmers, system developers or any other professional involved in the development of a help program.

The Help Design System (HDS) reflects an approach to job aiding that is innovative in both concept and application. Most guidance systems simply provide the developer with a list of discrete guidelines or rules to be used in the design. They pay little if any attention to how the developer can utilize the guidance. At best, they provide examples to aid the developer in understanding or interpreting the particular guideline.

However, design is not the serial application of independent guidelines. A help system is an integrated whole. The problem is not just instantiating a particular guideline in terms of a particular help design context. It is also the integration of guidelines that are often in conflict.

The strategy in HDS is to focus on aiding the developer to transfer the guidance to his or her application. There are two major components to the HDS. The first level consists of fully functional model help systems. The models are instantiations of not only the individual guidelines on help design but also the integration of those guidelines. The developer can enter a model help system, navigate through it, and get a complete sense of a good help system. The designer can then adapt that model (conceptually) to his or her own application.

We currently have one model help system -- help for a mock word processor called "Not Write". . However, a product ready for deployment would have several models each representing different sets of hardware and software constraints on the possible design. See Figure 1 for a visual representation of the Not Write online help system:

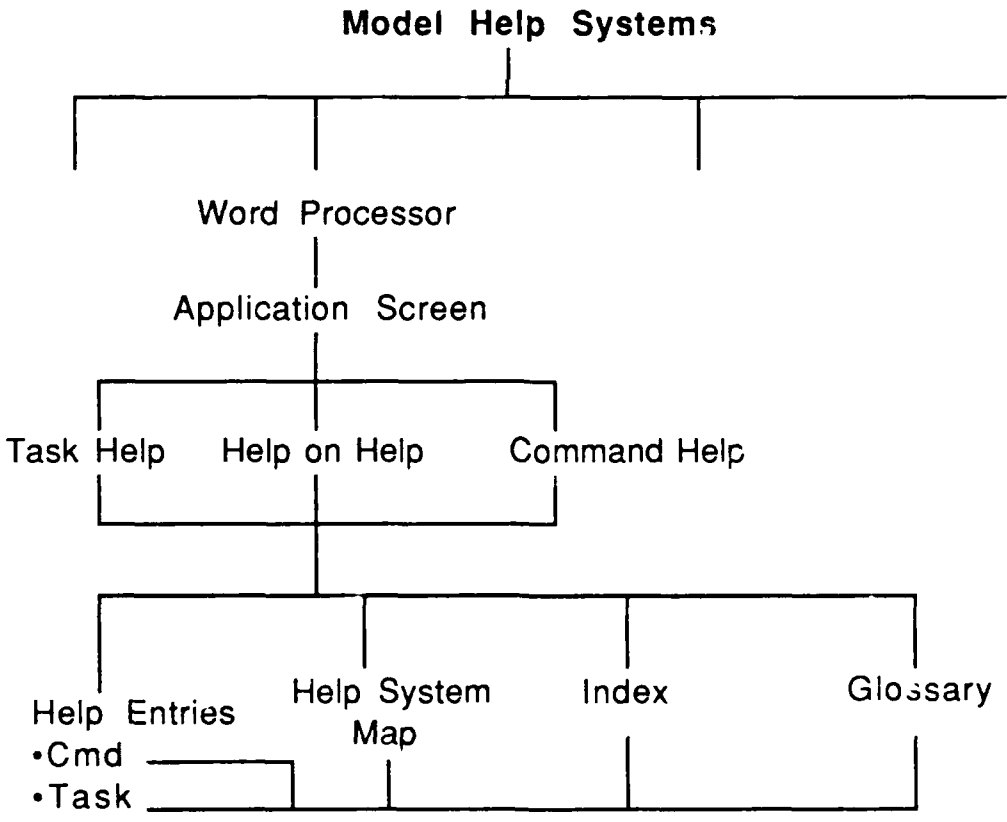


Figure 1. Components of the Not Write online help system.

The second level of HDS provides assistance in transferring or adapting the models. If the designer wants to see more explicit statements of the design strategy for any screen in a model, he or she can enter this second level for advice. There are a variety of aids provided as is shown in Figure 2.

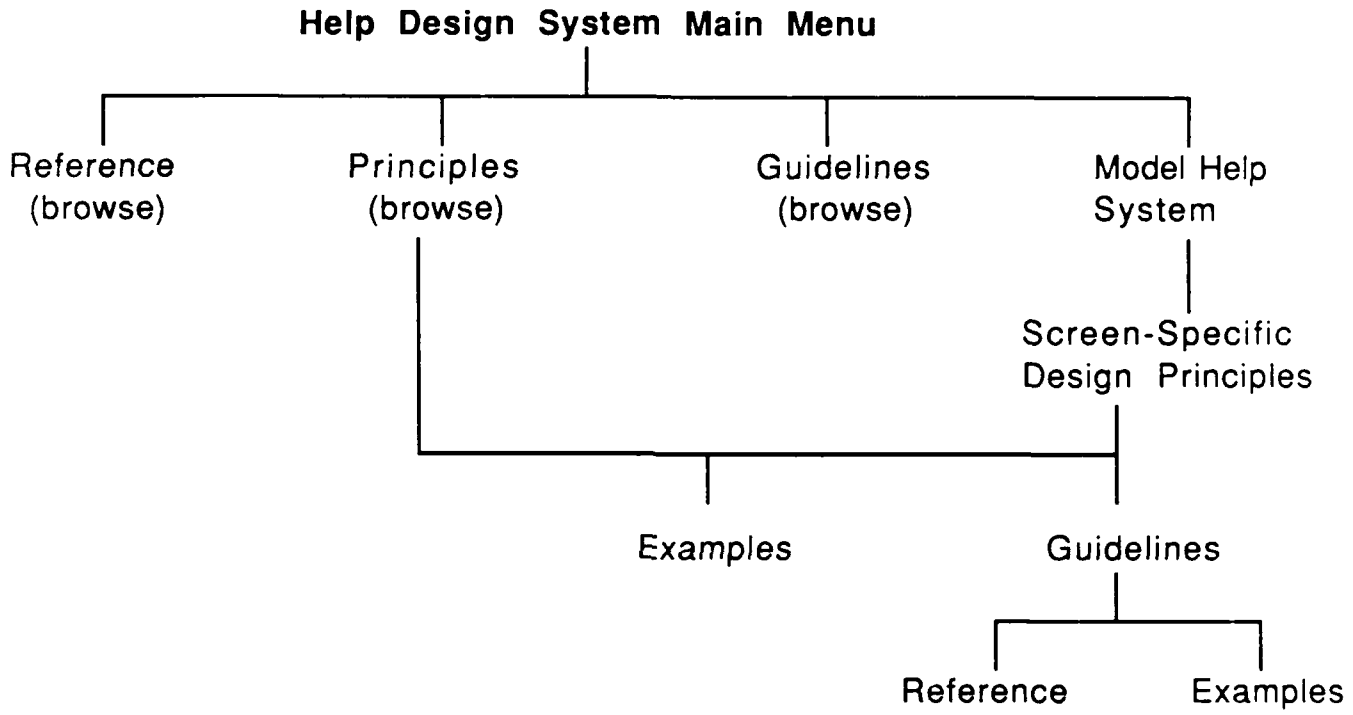


Figure 2. Components of the HDS system.

The idea is to provide more and more detailed information, ranging from an overview statement that basically points out the important features of the screen to more specific principles and other examples of those principles, and finally to detailed guidelines and even to references where the designer can go for more information.

The design aiding component now consists of 39 principles, 192 guidelines, and 198 references.

Using HDS. Includes brief written instructions on working with HDS. Other than that, the system should be self explanatory. Select "How to Use" HDS on the main menu.

As a part of HDS we have included a draft of a presentation on why online help is important. This may be accessed through the "The Importance of Online Help" selection on the main menu. This is meant to be a tool for HEL to communicate the importance and value of online help (as compared, e.g., to hardcopy documentation). We hope to have all the graphics for the overview completed by 1 April.

The Help Evaluation System.

This system is designed to aid the evaluation of existing online help programs. Thus, it is for use by end users of software as well as by software engineers. The goal in developing the evaluation system was to provide an objective and valid tool that would provide diagnostic information on the *strengths* and weaknesses of a particular help program as well as permit comparisons between help systems.

The need for the evaluation tool is obvious when one attempts to compare help programs or even describe a given help program. While we can talk about this or that feature of a help program, there is no system available for the systematic description of a help program. Since a given help program cannot be systematically described, help programs cannot be systematically compared. The comparisons are always on the basis of one or another feature.

Clearly, before we could evaluate a help system we had to be able to describe it. Thus the initial work on this project focused on a method for describing a help system. We used an information processing analysis of help to build a taxonomy of help tasks. That is, at the top level of our system are eight tasks that a person must do in getting help. The tasks range from formulating the help requirement and accessing the help system to comprehending the information and applying it.

Our criterion for a good help system is one in which the user can get and apply the help information in a minimum of time and cognitive effort. That is, a good help system minimizes the distraction from the primary task at hand -- using the application program. If a system minimizes time in completing each of the information processing tasks involved in getting help, then the system is maximally efficient. Figure 3 gives a summary of some of the key cognitive issues an effective online help system should address:

Information Processing Task:	Help System Feature:
1. Access help system itself	application interface
2. Select a potential topic	menu or keyword search
3. Locate information within a topic	format of text or screen
4. Understand help information	comprehensibility of text
5. Decide if information is relevant	type of information, content
6. (possibly) Go to *related topics*	navigation through help
7. Apply help information to situation	windows, link to application

Figure 3. Information Processing Tasks and Help System Features.

Thus, our approach to evaluation is to assess the degree to which a help system is efficient in supporting each of the information processing tasks. This immediately suggests performance testing to gather speed ad accuracy measures on a help system. However, if we are to achieve our goal of a tool that permits a comparison of help systems, performance testing is not feasible. The complexity of the application program being supported will affect the complexity of a help program. Thus, even the best designed help program for a large application will result in slower performance times and probably more errors than almost any help program for a small application.

We have turned instead to the use of ratings. Experts rate the help system in terms of particular strengths and weaknesses. The ratings are organized into

eight categories reflecting the eight information processing tasks involved in using help.

There are two evaluation instruments. The first one, the Help Design Evaluation, looks at the features of the help program that are provided to promote effective completion of each of the information processing tasks. We have completed two versions of this evaluation instrument and are in the process of collecting reliability and validity data on each one.

The second evaluation instrument, Help Content Evaluation, examines the accuracy and completeness of the information in help. The focus is the degree to which the information available will actually support the completion of tasks in the application program. This evaluation instrument is still in development.

Use of the Evaluation. Full instructions for the use of the Help Design Evaluation instruments are included with the rating scales. Over the next two months we will be applying these instruments to 25 help programs to provide a comparative database as well as reliability and validity data.

The Prototype Testing System.

The goal of this system is to facilitate the testing of different design strategies in an experimental setting. There will be two testing tools, a Menu Tester and a Navigation Tester.

The Menu Tester is partially developed and is currently supporting an evaluation experiment. The current form is a very specific tool. There are two menus in the system and three sets of menu cues (items to find on the menu). A full experimental design is implemented in which a subject receives cues and then makes the proper menu selection over a series of five blocks of 10 trials. The system collects all time and accuracy data in a format ready for submission to a statistical package.

In the full implementation, a developer will be assisted in putting the menus and stimuli of his or her choice and specifying the sequence of trials.

The Navigation Tester is still in the planning stages and the feasibility is still being assessed. However, the basic goal is the same as for the Menu Tester: a system that will support the design and data collection of experimental comparisons.

Use of the Menu Tester . This is really the presentation software for an experiment in progress. It is not a generalized tool. Use, outside of data collection, is primarily for demonstrating the Menu Tester concept.

Plans for the Final Year

The work for year three will consist primarily of completing the three evaluation and testing systems. This includes both completing the product development and elaborating upon the conceptual framework. We will discuss the plans for each system in this section. However, there is more general and futuristic planning that is also required.

In the conclusion of the report on the "Help Design Software" we discuss two possible directions that the HDS system could take in the future. Both of the considerations involved adding an intelligent component to the system. In one case it was to provide an intelligent job aid while the alternative was an intelligent trainer. A third alternative is to expand the definition of help encompassed by our system.

The HDS system was originally conceived of as a job aid that a designer could use in planning his or her help program. There are model help systems that the designer can adapt and there are help design aids to assist in that adaptation.

An extension of the system as a job aid could involve making it a more **intelligent assistant**. Under this conceptualization, the designer could work within HDS, modifying and adapting the model (or even the supporting examples) to fit his or her system. The design aiding system would then provide feedback on the quality of the adaptation, making suggestions, as appropriate, for modifications.

An alternative to job aiding is to adapt the system to the **training of help program designers**. In this extension of HDS, a model help system would serve as a simulation environment. The trainee would be given design tasks requiring either building a component of the help system or diagnosing and repairing "faults" in the help system. The simulator would, of course, be modified by adding the appropriate faults or deleting the system feature necessary to support the task.

The design aiding component would be the feedback system for the trainee. That is, the system could provide feedback on completed tasks (diagnoses or component building) or it could provide advice and hints during the course of the task.

The third alternative is to extend our work to **new issues in the design of help**. Our work focused on help as job aiding (aiding the end user, not to be confused with aiding the designer which is the goal of HDS). Our theory and the HDS system, could be extended to included the use of tutorials in online help. Extending the work in this direction would involve developing model tutorials as a part of (or alternative to) job aiding and, most importantly, developing principles, guidelines, and procedures for the design of tutorials.

Now let us turn to the more immediate future and present our plans for completing each of the design and evaluation tools.

Help Design Software.

We have been developing one model help system. We will **complete the development of that model.** This will involve adding components that are still missing : some command and task help text, the rest of help on help, and the map of the help system. We will not write help text for all commands and tasks. Since the model is only meant to be illustrative. (a model), we only need enough text to make navigating meaningful and to provide multiple examples of the content, layout, and style for presenting help.

We will also **test and revise the model,** to ensure that it really does provide a solid model of what help should be. We have already done some testing and know that the navigation system needs to be modified.

We will **complete the help design aiding system**. There are several components entirely missing from the aiding system: examples of principles, design strategies, and design procedures. These need to be added. We also need to add to the reference list and to the guidelines.

Finally, we will **test and revise the design aiding system.** The design aiding system is a level in the HDS, and it, in turn, has many levels. As a consequence we will need to ensure that the system has very distinctive guideposts and directions to support not only navigation but also the effective use of the design aids. User testing has already suggested many revisions we can make to aid the designer.

We will **complete the "How to use HDS"** component that is accessed through the main menu.

We will **complete the "Importance of online help"** component that is accessed through the main menu.

We will provide a **new introductory screen**.

Finally, we will be preparing more **technical reports** documenting the system and elaborating upon the theoretical underpinnings of the system.

Help Evaluation Surveys.

We will **complete the assessment of the reliability and validity of the Help Design Survey**. We currently have two versions of the Help Design Survey. The selection of the final version will be based on the reliability and validity assessment. Validity will be based on construct and content analyses. That is, the validity of the Survey will be based on the theoretical soundness of the approach and the adequacy of the particular items as judged by experts outside of our project area.

The reliability assessment will be done by having two expert raters use each instrument in evaluating 25 different online help programs. Interrater reliability will be calculated.

The reliability assessment will also **provide a data base of ratings** that can serve as a basis of comparison in applying the survey to other help programs.

We will **assess the feasibility of the Help Content Survey** approach to evaluating accuracy and completeness. If it is feasible, we will **develop a draft of the Help Content Survey** instrument and conduct initial testing of it.

Finally, we will be preparing more **technical reports** documenting the reliability and validity of the system and elaborating upon the theoretical underpinnings.

Prototype Testing System

We will complete the **development of the Menu Tester**. The final version will permit designers to enter menus of their own design (including hierarchical menus), enter stimuli to be presented, and specify a test sequence. In the final version, the end user will work through an experimental session totally presented in the Menu Tester, and accuracy and time data will be saved.

We will **assess the feasibility of the Navigation Tester** and if feasible we will develop a prototype system. The Navigation Tester, if developed, will have the same functionality as the Menu Tester.

Finally, we will be preparing more **technical reports** documenting the testing systems.

Instructions for Installing and Using the Help Design Software

by

Thomas M. Duffy, James Palmer, Brad Mehlenbacher,
Ann Aaron, Maria Truschel, Karen Denchfield, Alan Houser.

Communications Design Center
Carnegie Mellon University
Pittsburgh, Pa. 15213

1 March 1988

This work was funded in part by the United States Army
Human Engineering Laboratory, Aberdeen Proving Grounds, MD
under contract DAAA1-86-K-0019.
Contract monitor: Maureen Larkin.

You Have Received Three Diskettes

You require a Macintosh Plus, SE or II with a hard disk to run the Help Design Software.

Installing the Help Design Software

To install the complete Help Design Software package on your hard disk, follow these instructions step by step:

Use the "New Folder" command under the "File" menu to create a new folder.

Name the folder HDS.

Insert the "HDS Help Design Software" System Disk into the Macintosh disk drive.

Copy the contents of the disk "HDS System" into the folder "HDS" which you created on your hard disk.

Eject the disk "HDS System" by dragging it into the trash can.

Insert the disk labelled "HDS Help Design Software #1" into the Macintosh disk drive.

Copy all the files in the disk "HDS #1" into the "HDS" folder on your hard disk (the "Select All" command in the "Edit" menu is helpful to use when doing this).

Eject the disk "HDS" by dragging it into the trash can.

Insert the disk "HDS Help Design Software #2" into the Macintosh disk drive.

Copy all the files on the disk "HDS" into the folder "HDS" on your hard disk.

Eject the disk "HDS #2" by dragging it into the trash can.

Choose the "Restart" command under the "Special" menu to restart you Macintosh.

Running the Help Design Software

You are now ready to run the software: open the "HDS" folder and double-click on the file "Home".

Do not click on any other files. The Home file contains all the major functions used by all the other HDS files.

A Brief Summary of the Help Design Software

Once you have started the program, an introductory screen will appear. Click once anywhere on the screen and you will be taken to the credits page. Clicking on this page takes you to the "Help Design Software Main Menu".

The Main Menu should give you the best sense of the functionality of HDS. After making a selection you will be taken to one of the various components of HDS. And the Main Menu is always with you,

Note that in the Model Help System (which is a mock application supported by a "good" and functioning Online Help System) the HDS functions are not available to you; in order to return to the HDS support functions, choose either "Quit" under the "Control" option or "Map of HDS" under the "Help" option.

The first two choices are "The Importance of Online Help", which describes online versus hardcopy help presentation of guidelines, and "How to use the Help Design Software", which gives a short description of each component of the software you are using. These should be reviewed first, and should give you an overall view of the software.

The third choice is "Map of the Help Design Software" which is both a visual representation of the overall system and a navigation aid (you can click on any part of the map to go to that component of the system).

The fourth choice, "Examine Model Help Systems", takes you to the Model Help Systems Menu. Select the "Word Processing Application" to explore a functioning help system for our mock word processor, Not Write.

You can also select the "Browse Guidelines", "Browse References" and the "Browse Design Principles" to view the extensive support material behind the building and presentation of the Model Help System.

The Help Design Software is designed for use without extensive hardcopy documentation. If you have any questions or concerns feel free to contact be Tom Duffy at the Communications Design Center, Carnegie Mellon University.

Running the Menu Order Experiment

Open the "HDS" folder and double-click on the file "Menu Order".

You will be presented with an introductory screen and a menu bar at the top of the screen. This screen is where you choose the condition you want to test. The two selections, MenuType and TaskType, are the main experimental variables: there are two types of menus, alphabetic and functionally organized, and three types of tasks, direct match (find bold on a menu with

the item bold on it), synonym match (find heavier on a menu with bold on it) and picture match (find a before-after representation of bold on a menu with bold on it).

Make a selection from both the MenuType and TaskType menu bars, and the experiment will begin.

To quit the "Menu Order" program, press Command-Q at any time.

See the paper "Finding Menu Information: Linking Menu Organization to the User's Goals" for a detailed description of the theoretical issues being addressed by the "Menu Order" software.

Developing the Help Design Software

by

Thomas M. Duffy, James Palmer, Brad Mehlenbacher,
Guojun Zhang, Ann Aaron, Maria Truschel, Karen Denchfield

Communications Design Center
Carnegie Mellon University
Pittsburgh, PA 15213

1 March 1988

This work was funded in part by the United States Army, Human Engineering Laboratory, Aberdeen Proving Grounds, MD under contract # DAAA1-86-K-0019 AC85.
Contract monitor: Maureen Larkin.

The Help Design Software was developed for use in the software engineering environment to aid in the design of online help systems. We define online help as a program and database that is a resource for using an application program or some piece of hardware (Duffy and Langston, 1985). That is, it is a program that, at a minimum, permits access to and navigation through a database. The help program and database are functionally distinct from the application software and access to help is initiated by the user.

Online help is distinguished from error correction or error feedback in that error correction is typically a part of the application program, rather than a separate program and is system, rather than user, activated. Thus, online help is a computer-based information system accessed by a soldier and designed to provide the soldier with assistance in completing a task within another computer program.

The goal for online documentation is job aiding; it is a source of reference information and procedural instructions relevant to specific tasks or jobs. We distinguish online help from tutorials or training on the basis of this job aiding vs instructional function. The user of online help is seen as a soldier in the middle of a job rather than a soldier learning about a job. (Intermediate needs vs. long-term needs.)

We do not distinguish between online documentation and online help. We will classify a system as online help as long as it is a functionally distinct program and database designed to aid job performance. If a technical manual that was designed to aid a soldier in using a computer system were placed online, we would classify that online manual as an online help system. We would hasten to add, however, that we would expect such an online help system to be ineffective: the design of an online help system is quite different

from the design of a technical manual (Duffy, Gomoll, Gomoll, Palmer, Aaron, 1988).

We are using "software engineering environment" in a broad sense to include all efforts to develop application software. The intended users of this design tool are contractors or DOD personnel involved in the design or evaluation of a help system for application software. Because these users come from a variety of disciplines, we make few assumptions about their skills and knowledge. A user may be a technical writer, computer programmer, systems engineer, or a human factors specialist. More than likely the online help designers will be a team consisting of personnel from several of these disciplines.

Existing Strategies for Aiding

Guidance on the development of online help, technical manuals, or the user interface is typically provided in the form of a series of verbal statements of detailed design and content requirements: the ubiquitous "guidebook" or "specification." There is a long history of presenting these discrete chunks of verbal information as a means of guiding the writing process, as is evident in the examination of almost any guidebook for writing, e.g., Strunk and White (19). It is also commonplace in guiding interface design, as can be seen in the comprehensive guidebook recently developed by Smith and Mosier (1986). Indeed, the Department of Defense exerts control over its contractors through these discrete verbal statements presented in the form of military specifications and standards.

While there has been an insistence on developing guidebooks, there is a vast array of evidence indicating the failure of this approach to aid the development process. Perhaps the most intuitively appealing evidence is the continuing difficulty in producing effective documents in spite of the availability of guidelines and the imposition of specifications. More specific

evidence comes from a survey by Duffy, Post and Smith (1987) of publication houses producing technical manuals for the Department of Defense. They asked writers and managers to list the problems they confronted in producing effective manuals. Their difficulty in understanding and applying the specifications was near the top of that list.

In part, the problem with guidelines is that they are generalizations that must be interpreted for the particular instance. The concepts in a guideline tend to be stated abstractly, and thus the interpretation requires skill and experience. If the guidelines were accompanied by training in which the writers practiced discriminating between the proper application and violation of the guideline, then they could develop the ability to apply the guideline in new situations (see, e.g, Merrill and Tennyson, 1979).

Developers of guidelines have attempted to overcome the problem of instantiating guidelines by providing examples. The Department of Defense Handbook 761, for example, presents examples of each of the guidelines. While this may help in applying a given guideline in isolation it does not assist in the coordinated application of the guidelines. Guidelines are a decomposition of a text. That is, while a text is a whole, guidelines function only as a series of independent statements about that text and cannot possibly capture the interaction of principles and design strategies that resulted in the final product.

The fact that guidelines cannot capture the essence of good documentation is reflected in Klare's (unpublished) analysis of fifteen different guidebooks. He found that there was seldom agreement between the guidebooks; different writers viewed the critical attributes of good documentation differently. More importantly, Klare found many conflicting guidelines within any given guidebook. As a simple example of that conflict, many guidebooks advise writers to "be concise" and also "be complete." These guidelines are clearly in conflict: when the inclusion of a piece of information is debatable do we include it to be complete or do we exclude it to be concise? It is in the

resolution of these conflicts and integration issues that lists of guidelines, even with examples, fail.

Rationale for the HDS Approach to Aiding

Our approach is based on two considerations. First, the same principles for the design of a help system apply to the design of our Help Design Software. The designers are very much goal-oriented and want help with a minimum of distraction from the design process.

Second, people learn most readily from modeling. They don't use manuals (text) to assemble a bike, repair a car, write to a specification, or design a display. Rather, they look for a model to imitate. They look at another car, they find another manual written to the specification, etc. (see, e.g., Kern 1985).

These factors both suggest that the most effective method for aiding designers is to provide a model. Examining a model is less distracting then reading text on guidelines or design procedures. It also provides an integration of the design principles more nearly approximating the goal of the user, i.e., a help system. Thus while we focus on the use of examples, it is on example help systems not examples of guidelines. We thereby attempt to bridge the problem of interpreting and applying conflicting advice as well as abstract advice.

While we are attempting to aid job performance (i.e., designing a help system) the basic psychological goal is one of learning. That is, the use of guidelines, the HDS or any other aiding system is to help the individual develop a representation of a well-designed document and the conceptual skill to generate such a document. An examination of instructional theories (both cognitive theory and the more traditional behavioral theory) provides very

strong support for example based learning. In cognitive science, the use of examples and analogies is seen as essential to efficiently building a representation or schema of the problem domain and the solution strategies (Glick and Holyihok, 1985; Anderson, 1987).

In behavioral theories of instructional design, also, there is no question as to the importance of examples. The only controversy has been the proper sequencing of the presentation of the concrete examples and the abstract principles or concepts. Presenting guidelines without examples aids the person in remembering the guideline, i.e., remembering the verbal statement of the guideline. Aiding an individual in using the guideline requires the availability of examples of its application.

As we noted above, if we want the individual to use the document design principles then we must present examples of effective documents -- not of isolated guidelines. Spiro (1987) has recently made a strong case for an instructional approach that is based on presenting complex, integrated examples. He argues that almost any knowledge domain is ill structured when it is considered in the context of real-world applications. By ill structured he means that there are no algorithms or direct paths to the solutions of the problems (to a good design). This lack of a directly reached solution is exemplified in medical diagnosis, economic forecasting and other complex domains, domains in which we find that instruction is based on analyzing case studies. However, the issue also arises even in learning or applying a simple concept like a game. Imagine trying to tell a creature from outerspace what a "game" is so that it can view any event and classify it as a game or not a game. The task is not possible since there are no clearly-defined attributes of a game that will apply to all instances of games. There are a wide variety of attributes of games and only some subset of those attributes will apply to any particular game. Any particular attribute may or may not be a relevant feature of a given game. Learning the list of possible attributes will therefore not lead to the ability to recognize a game.

Spiro's argument applies to document design; for the reasons discussed above, guidelines cannot be applied directly in the real world of writing. They must be interpreted in the context of the particular application and they must be integrated (including the resolution of conflicts between guidelines) to produce a final design. Just as games may be competitive or cooperative, a particular segment of a document may be concise or complete. Presenting guidelines with examples is much the same as presenting individual characteristics of games with examples of games where the characteristics apply. That sort of advice cannot lead to the learning (and hence the production) of what constitutes a good document any more than listing all the possible features of games (include competitiveness and cooperation) can lead to an understanding of games or the ability to produce games.

Spiro argues that the development of the ability to transfer knowledge in ill structured domains, such as document design, will be supported most effectively through case based examples. That is, the focus is on the presentation of complex, real world cases that represent the integration of the principles and guidelines. Principles and guidelines are presented in the context of the cases. Not all principles will apply to all cases nor will they apply in the same way to each case. In this way, the learner can see instantiations of the principles as well as the integration of, and tradeoffs between, principles.

Our approach to aiding online help designers follows very closely the instructional model presented by Spiro. The top level of the aiding system consists of model help systems that the designers can examine and use. The systems are fully functional and thus instantiate the integrated application of guidelines. However, the aiding system is not based on the use of models or cases alone, as that would not provide assurance that the designers could transfer the proper principles to their help systems. The models do present the integration of design principles, but they do not assist the designers in focusing on and understanding those principles..

The second level of the Help Design Software (HDS) provides the principles and guidelines that are presented implicitly in the model. For any one screen of a model help system the user can obtain the principles and guidelines that supported the design of the screen and can obtain additional examples of the application of the principles. Thus, while the focus is on an integrated instantiation of a help system, there is a support structure to provide the guidance for implementing the design strategy.

In summary, we see that the most effective way of aiding the help design process is to provide model help systems that represent the integrated application of sound design principles. Ideally, the model will closely approximate the constraints of the developer's help system. That is, the model will reflect help for an application of the same degree of complexity and delivered under the same general hardware and software constraints that the developer is facing. To the extent that this is the case, a developer can easily adapt the model help system to his or her own application.

To the extent that the model does not represent the constraints of the developer's application, he or she may have difficulty transferring the principles embodied in the model. Therefore, principles, guidelines, development procedures, and examples of principles are provided to further support the development process.

The HDS Design

The model help systems.

At the top level of HDS are several fully functional help systems. These help systems are meant to serve as models that designers can adapt to fit the requirements of their specific application.

A critical design feature of the model component of HDS is that at least one model closely approximate the constraints of the designer's application. There are at least two implications of this design objective. First, the help systems should not incorporate elaborate or complex design features. We want to present design features that designers can actually adopt, not features they can only wish they had the time, money, or system capabilities to incorporate. Thus, we will not look to natural language mediation, collecting elaborate user histories, incorporating artificial intelligence, etc. Rather, we want "model A".[1]approaches to design; good, solid and basic designs that are easy to use.

The second implication of the design objective is that a designer should be able to find a model that approximates the constraints under which he or she is developing a help system. These constraints are characteristics of the content and complexity of the application, the capabilities of the delivery system, software, and the features of the delivery hardware that will significantly impact the deign of the help system. In essence the design of this system of models requires us to answer the question, "How many basic help system designs are necessary to reflect the essential features of the variety of help systems in use?" Alternatively, the question might be phrased as asking for the number of kernel or primitive help systems that are necessary to reflect the universe of help systems. We see this as a question, or perhaps a philosophy, that guides our design of HDS. We do not see it as a question for which we will provide a comprehensive answer.

Key to the consideration of the number of kernel help systems is an analysis of the number of system constraints that will affect the design of a help

When we refer to "accomplishing their goal" or "using the application software" the user may be in the actual execution of a plan or he or she may be developing a plan. It makes no difference. The important point is that they need assistance in bringing the plan to an acceptable state (I say it that way because he or she may be looking for a more efficient plan or to use a particular system feature.

---

[1] We thank M. David Merrill for this metaphor for the design strategy.

system. Our earlier work, for example, found that screen size is a critical feature in the design of help systems. Developers working at large screens tend to focus on the development of continuous text that a user scrolls through. Developers working on small screens, in contrast, emphasize the use of brief, highly formatted help texts that a user pages through (Duffy, Gomoll, Gomoll, Palmer, and Aaron, 1988).

We would also expect the characteristics of the application program to significantly impact the design strategy. For example, a program with a large number of commands will require a menu hierarchy whereas a small program can work from a single menu. A program that is a general tool used for many tasks may require more conceptual and heuristic presentations in the form of prose whereas a very specific application program can be supported by more procedural instructions. Finally, a program that has many modes (or uses), .e.g., a spread sheet, can best be supported through context sensitive help while a modeless system, e.g., a simple editor, cannot be supported with context sensitive help.

This issue, the constraints on the design of help systems, will be a continuing conceptual focus in our work. At a more practical level, we are beginning with the development of one model help system. The help will be for a mock word processor that has a total of sixty commands that support editing and formatting. The delivery system is a small screen (a Macintosh screen) with a mouse, menus, and overlapping windows.

The design aiding system.

The design aiding system is meant to provide two types of support. First, it can be used by itself as a resource of principles, guidelines, and references to the literature on the design of online help. A designer can use the design aid resource to review design issues or to locate relevant discussions of those issues in the literature.

The second use of the design aiding system integrates it with the model help system, allowing the designer to request guidance that is specific to any screen of the model. We see this as the conceptually more important application. It reflects our view of how to give advice effectively, and it is a step toward an intelligent advisor or analyzer.

In this approach to aiding, the HDS system helps the user understand and transfer the design strategy presented in the model help system. In essence, while exploring the model help system, the designer can ask for advice on the strategies used in developing any one of the screens in the model. The user then enters the aiding system not as a "browser" free to roam, but rather in a mode that will provide principles, guidelines, examples, and reference relevant to the particular help screen.

The aiding system is hierarchically organized to provide more and more detailed advice. That organization is illustrated in Figure 1. If the designer asks for advice on a particular help screen, he or she will first see a narrative description of the design strategy that governed the design. The narrative will focus the designer's attention on the critical design features and present the rationale for those features. It also describes alternative designs that were considered and rejected. Thus, this top level provides a general rationale.

Given the general rationale, the user then has two options. First, he or she can ask for any methodologies that were used in achieving the design. For

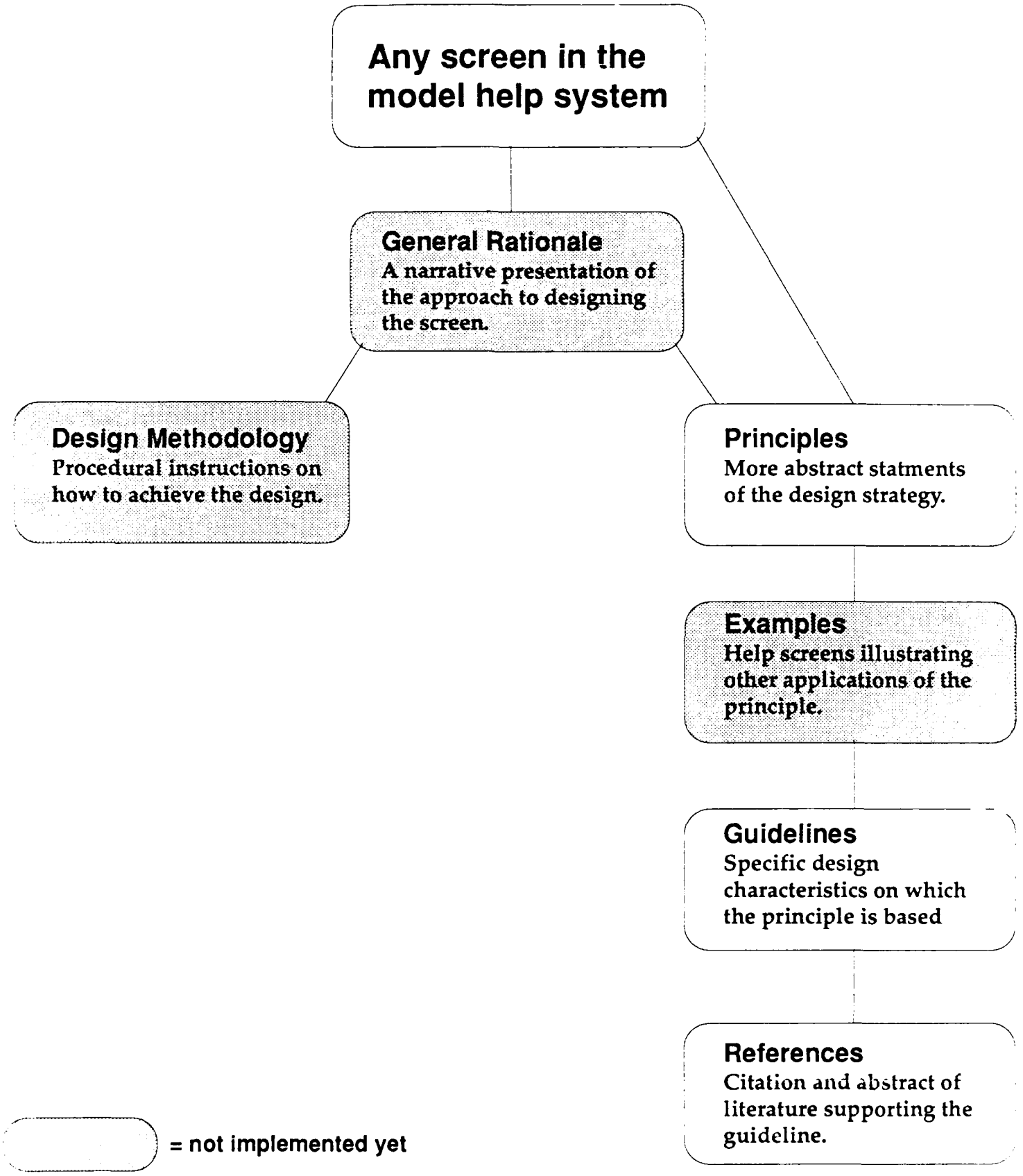


Figure 1: Hierarchical Organization of the Design Aiding System

example, if the designer is asking about a menu screen and the strategy was to provide a functionally oriented menu, the methodology support will provide the designer with the steps for obtaining a functionally organized menu.

The alternative path from general rationale is to ask for the specific design principles that governed the design of the help screen. The principles are basically more abstract, or generalizable, statements derived from the general rationale. If the constraints on the designer's help system are considerably different from the constraints underlying the model, the designer may have difficulty transferring the general rationale to his or her application. The principles, being more general statements, are meant to aid the transfer process for this situation.

In addition to this, there are also examples (not implemented yet) which are full screen layouts that reflect the application of that principle. In generating examples, we have tried to provide widely varying contexts to illustrate the principle. Thus, the content, the complexity of the system, and the particular strategy for applying the principle will vary between examples. In this way the designer may find an example more closely related to his or her application or, at a minimum, will be better able to understand the principle (see Merrill and Tennyson, 1979 re the presentation of examples).

There is also supporting information accompanying the statement of each principle. The category box tells the user what taxonomic category the principle is concerned with (that is, did the guidelines on which this principle is based have to do with perception, decision making, understanding, motor activity, or system response time?). The principle number box, appropriately enough, tells the user the number of the principle that he or she is on. This orients the user as to his or her current position in the Principles database relative to the first principle.

As well as the examples for any one principle, the designer can also ask for the guidelines on which the principle is based and which are relevant to the help screen. The guidelines were gathered through a review of the literature on interface design, online help design, and document design. Being more specific statements of what the help system should look like, the guidelines are meant to help the user interpret the principle and thus more readily apply it to his or her application.

The database of guidelines captures the following information:

- the guideline or rule itself;
- the rationale for following the guideline (specifically, does it affect perception, decision making, understanding, motor activity, or system response time?);
- the information processing task (referred to as a "taxonomic category" in the guidelines database) to which the guideline applies. That is, is the user involved in accessing the help, characterizing the task, moving around in the help, easily reading the help, finding all the necessary information in the help, being able to understand the content, or quickly applying the information to his or her problem? (In the guidelines database, these processing tasks are labelled: access, menus, navigation, format, content, comprehensibility, or link to the application.);
- whether the guideline refers to the evaluation of the surface appearance of the help system (referred to as "product" in the guidelines database), or to the act of producing some aspect of the help system (referred to as "process" in the guidelines);
- the design parameter to which the guideline specifically applies (screen format, text density, etc.);
- the basis on which the author has formulated the guideline (either personal research, cited research, or observation);

- additional comments by the HDS designers about the guideline or about the source from which the guideline was gathered.

This information is presented entirely on one screen in individual boxes. The user may click on these boxes to get information about what the information labels like basis, product, and process mean. As with all components of the Help Design Software, the guidelines database is linked to the database of reference citations, as well as to the more general levels of HDS like the design methodology and the principles.

The designer can ask to see the references on any guideline by choosing the appropriate menu option. The request will take the user to the relevant set of references and provide him or her with the full reference citation and an abstract of article as it is relevant to the design of online help.

The system as described above is focused on aiding the designer in transferring the design strategy reflected in a particular screen of the model help system to his or her own application. The designer will only see the principles relevant to the particular screen in the model, the guidelines that are applicable to both the help screen and the principle, and the references that are relevant to the guideline.

The alternative mode, browse mode, for using the aiding system, provides more open access to the information. The user in browse mode can enter principles, guidelines, or references and freely browse through them. There are search and sort capabilities for each of these sets of material. Thus the designer can sort on some characteristic of set of material, e.g., sort the guidelines or principles in terms of the user task that is most relevant, and then examine all of the principles or guidelines relevant to any particular task, e.g., navigating. Alternatively, the designer can use a keyword to search, for example, for a reference by a particular author or for guidelines on a particular screen element.

The design strategies and design methods are not available in the browse mode since they reference particular screens in the model. Also, the examples can not be browsed freely. Since the example is only meant to illustrate a particular principle, the examples can only be accessed through the principles and only examples on the particular principle selected can be viewed.

Summary

The HDS system provides formal guidance to the designer by providing a model help system that, more or less, approximates the design constraints under which the designers help must be developed. Transferring or adapting the model to the designers application is seen as the primary approach to aiding.

A hierarchy of design aids are available if the designer has difficulty in interpreting or transferring the design principles. The aiding system is individualized to each screen in the model help system so as to provide screen (or component) specific information. The hierarchy ranges from a narrative presentation of the design strategy and more formal statements of principles to references in the literature that support a particular aspect of the design strategy.

In addition to this formal guidance, the HDS can also be used as a resource for the designer. He or she can browse the model help systems, principles, guidelines, and references in support of planning a particular application or as a checklist in evaluating work accomplished.

Current Status of the HDS System

The Not Write help system.

The first prototype or model help system is for a word processor we have titled "Not Write". A beta version of the architecture for the help is complete and approximately 80% of the content is written and in place. Not Write is an EMACS-like word processor, presented on a small (11") screen. It has menu bars and commands can be issued through either keystrokes or menu selections.

The goal in designing the help for Not Write was to present a model help system. Therefore, there is no reference to the design aiding system in the help. Rather the designer is notified upon entering the model that he or she should press "control ?" any time information on the design strategy is desired.

We will now review some of the features of the help system. The review is organized in terms of the tasks a user must perform in using a help system. (See the Theory and Methodology for Evaluating Online Help report included in this binder for a discussion of the task analysis).

Formulating the help requirement. The user must first formulate a statement as to just what help information is required -- what he or she will search for in the help system. Experts will likely formulate their needs in terms of specific commands.. However, other users may think in terms of the real world task they are trying to do or in terms of some computer task. Furthermore, the users may or may be using the terminology as it is used in Not Write.

Most help systems require the user to use the system terminology, i..e, the user must adapt to the semantics of the computer. However, an effective

help system must be able to capture the users expression of needs. In help for Not Write we have provided two support systems. First, we have a task menu in which the entries are not computing tasks but rather the real world task the user wants to perform. That is, the menu reflects editing and formatting tasks as a person in a noncomputer environment would characterize them, e.g, comparing two documents.

The second support is a keyword search capability (not yet implemented) that includes a wide variety of synonyms for commands and tasks. Since the keyword list is not shown to the user, it can contain extensive fully and partially redundant terms reflecting the names novices, experts, and transfer user might give to tasks or commands.

Access. Given the formulation of an information need the user must access the help system. Some programs make accessing help unbelievably difficult, e.g., one system we reviewed required the user to type "control -]" to enter help (Duffy, Trumble, and Palmer, 1988).

The user of HDS enters the Not Write model on a screen of Not Write with some text on the screen. Thus the designer enters the model help through the mock application. We have placed help as an item on the Not Write menu bar so that it is always visible. Furthermore, there are three alternative selections one can make from that menu, that help place the user in the most relevant area of the help system.. Two of the access choices are to alternative menus in the help system. The user can choose to go to a task menu or to a command menu. The third access is through a previously specified bookmark in the help system. If the user has been in help before and there is a particularly difficult or important topic he or she can mark that place with a bookmark. Then, any time they need access to that information while working in Not Write they can simply select the bookmark and go directly to the relevant screen.

Topic selection. Once in the help system the user must select a topic to examine. The help provides both task and command menus to support that selection. The menus were designed to emphasize breadth rather than depth. That is, we attempted to avoid creating menu hierarchies as much as possible. If all of the commands could fit on a screen they were placed there. If not, we used pop up menus attached to items on the main menu. Thus a user can search for the relevant topic without leaving the main menu. Also, avoiding a menu hierarchy also keeps the user from going down a garden path (making several selections to get to help only to discover it is not the right information) and having to retrace and potentially get lost.

The menus are organized into groups of related commands or tasks to support the user's search for a semantically relevant entry. The command menu mimics the structure of the Not Write command menu thus supporting learning of the Not Write organization. Selection of an entry on the command menu leads to command help.

The task menu is organized into related kinds of tasks and the subsets of tasks have a category title (the titles are not yet implemented).

The index option in the Help System Functions gives the user an alphabetical listing of the application program commands which the user then may click on to get help on a particular command.

Finally, a keyword system, accessed through the search for button, serves as a back up selection strategy that a user can employ when a relevant item cannot be found either on the menu or in the index.

Format. The next information processing task for a user accessing help is to scan the screen on which the information is presented. Here, issues of layout, design, and readability are of importance. HDS addresses these issues in

several ways. Users are presented with a help screen which is esthetically pleasing–and the bold title area placed on a medium gray background reduces eyestrain as well. Examples and explanatory text are presented in boxes that "pop up" when the user clicks on a prompt, thus preventing screen clutter, and at the same time allowing a great deal of information to be accessed from one screen. In the same vein, permanently displayed information boxes are scrollable when it is necessary to display more information than can be contained on on screen. Typefaces such as Palatino and Athens were chosen for their legibility on screen.

Content. Once the user has reached the information in the help system, several questions arise: Is the information presented in enough detail to answer the user's question? Is the information presented in context? Is the information geared toward job-aiding (i.e., task-oriented)? The HDS answers these concerns in several ways. Within the help, users can click on an examples option to access an example of the particular command or task they are seeking help on. This example presents, in a real-world context, how the command is used and what the result will be. In addition to this, the example aids the user in accomplishing his or her task because it shows exactly how to implement the command.

Users also have the choice of seeking even more information when they are in the help system by clicking on such options as defaults, which tells them about the default settings of the particular command; warnings, which alerts users to potential problems when using the command; related commands, which give users a chance to recover if they mistakenly chose a command which did not quite meet their needs but a related command would work; read me, which alerts users to any changes or updates that have been made to the command; and send a comment, which allows users to send comments and suggestions to their system administrators (not implemented yet).

Comprehensibility. Assuming that the user has characterized his problem, accessed the help, located the first level of information and within that,

located information specific to his problem that will aid him or her in completing some task, the next step for the user is to understand what is presented. This is accomplished in HDS by writing that follows the conventions of style and grammar as well as reflecting the current thinking on technical and procedural writing. To ensure understanding for users who may be less computer literate, a glossary option (not implemented yet) will take the user to a database of definitions for the application program commands.

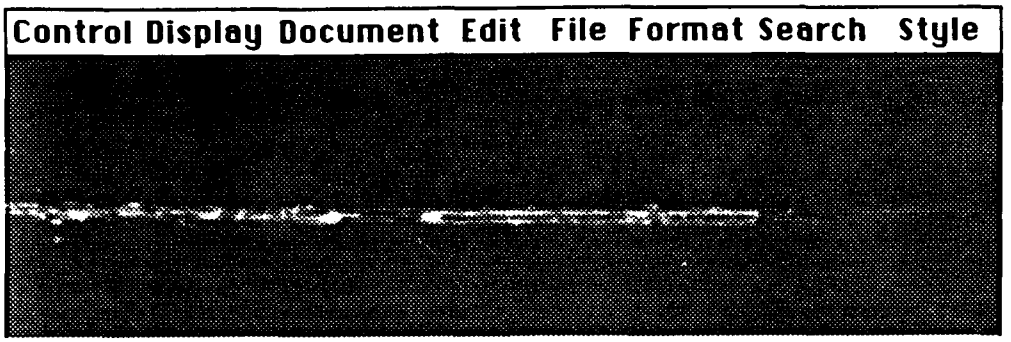
Navigation. Navigation, or the method of moving from one place to another, in the HDS is still in the development stage. The Help System Function options at the bottom of the screen provide several methods for moving about in the Not Write help system (see figure 2 for an example of the appearance and placement of the HSF options). These navigation options are always displayed while the user is in help. Thus he or she can choose to go to a different menu, to help on help, the glossary, the index, toggle to Not Write, go to the previous help screen, or exit help. These options can be selected at any and from any place in help. The options specifically related to moving around within the help system operate in the following way:

- **using help** provides the user with information about how to use the help system to locate information. It describes all of the features of the help system and how to get around in the system.

- **map** provides the user with a map of the application help system. The user may click on any portion of the map to go to that section of the help. If the user is in the design aiding portion of HDS, the map will show the entire HDS system.

Help on the Not Write™ Commands

HDS Map



Pull down a menu item and choose a command to receive help on.
The menubar works just as it does in Not Write.

Go to other menus or use help features

Help System Functions						Help Menus
using help	glossary index	map search for...	bookmarks toggle	Go Back	quit help	Not Write™ Tasks Not Write™ Commands

Figure 2:
Placement of Help System Function Options

Link to application. After the user has read the information in the help and understands it, the next information processing task he or she must attempt is to transfer the knowledge to the problem at hand in the application program. Because information such as command syntax, or how to implement a command is stored in short term memory, it is essential for the user to transfer this information as quickly as possible before it is forgotten. The HDS handles this requirement with two Help System Functions:

- **bookmarks** lets the user mark the place where he or she "left off" in the help system, and return to the application program. When the user reenters help, it will be at the point where the bookmark was placed.

- **toggle** provides the user with a way of quickly switching back to the place in the application program where she "left off" to access the help. Unlike bookmarking, it does not mark the place in the help system, when the user reenters the help, it will be at the beginning.

The design aids.

The design aid system is outlined in figure 1 and discussed in the previous section. The details of that discussion will not be repeated here. In this section we will describe the status of each of the information sources.

Principles. We have developed 32 design principles. We think that the list is complete, but we will continue to seek input on more principles. The most relevant information processing task that the principle support is included with the statement of the principle and is a basis for organizing the principles.

The user can ask to see the guidelines relevant to any given principle by selecting that option on the menu bar. If the user makes this choice, he or she

will be presented the first of the relevant guidelines and a toggle to jump to each of the other relevant guidelines before toggling back to the principle.

The user can also ask to see examples of the principle. The movement through the examples is analogous to the movement through the guidelines.

Guidelines. We have a list of 192 guidelines. This list was developed through our own design knowledge and beliefs as well as through a review of the literature on document and interface design. For each guideline, we identify the sources in the literature that proposed the guideline and we note whether it is supported by research, observation or expert judgement. We also specify the interface feature and the user task to which the guideline applies. These classifications are supplied to support the designer in searching for guidelines on particular issues.

The guidelines can be searched for particular information or they can be sorted on a variety of features to support the particular review goals of the designer. The sorting and searching is accomplished through menu selections using the top menu bar.

The guidelines are also linked to references and principles. A designer can ask to see the principle from which this guideline is derived (not yet implemented). When this option is selected, the system will toggle the user to the principle and then back to the guideline.

The user can ask to either see the references relevant to the guideline or can simply ask to move to the references to browse them . If the former choice is made, the system will take the person to the first of the relevant references and then permit him or her to toggle to each of the other references and finally back to the guideline.

References. We currently have 198 references online. However, there is still considerable updating and editing of the references to be done. The references provide a full citation and an abstract. Users view one citation at a time.

The references can be sorted and searched in the same way as was described for the guidelines. The sorting can be done by author, source, or year. Searching is by keyword and any occurrence of that word will be identified.

The citations are cross referenced (partially implemented). If a reference is cited in the abstract of a paper, the user can click on that reference and, if it is in our system, he or she will go to it.

Examples. Not yet implemented.

Design Method. Not yet implemented.

Design Strategy. Not yet implemented.

Authoring

The model help systems are designed for browsing only. Users can not add to, copy, or otherwise modify the models.

The design aid system has two modes: browsing and authoring. Normal use is in the browse mode. That is, the user is simply using the information provided. However, by resetting the user level to 5, the user can enter the authoring mode. Authoring will only apply to principles, guidelines and references. When the user is in author mode and in one of these domains, there will be a new top menubar. The menu will provide the ability to add

new items in the domain or delete existing ones. When a new one is added, the basic form appears on the screen and the individual simply types the specific information. Standard Macintosh procedures for entering text apply, e.g., the user clicks in at the place where he or she wants to add information.

Existing information can also be modified while in the authoring mode. If the user is in authoring, the cursor will change to the text icon (I) when it is in a region where text can be entered. When the text icon appears, the user simply click the mouse at the particular place where text is to be entered and begins typing.

Conclusions and Plans

We have tested the Help Design Software with six professionals using a Macintosh II with a 40 Mb hard disk. The tests last about 90 minutes and consisted of a mixture of tasks the subject was given to complete and requests for open-ended review of the system. The professionals included an interface designer from Symbolics Inc., a developer of help systems at Oracle Corp., a human factors/ interface designer, a cognitive scientist, a technical writer, and a designer specializing in interface design.

The reviews were generally very favorable. There was unanimous agreement that this was an excellent approach to aiding designers. The subjects found the help system easy to use and reflecting sound design principles.according it high praise. However, it was clear that more models are needed to reflect the variety of system constraints these subjects encountered in their work.

While the help design system received praise, it was also the part of the system that users had the most difficulty with. the problems were mainly in terms of terminology (e.g., the category names on the guideline cards) and

navigation ("going back" and understanding options at a particular place presented considerable difficulty). We are currently working at revising these aspects of the system.

We hope to have completed the Help Design Software, with one model, by June of 1988. We see two possible paths for future work with the system. First, the approach represents a new and exciting model of job-aiding strategies. We could continue to develop that aiding model. This would include the study of the help design process in which we would work with professional help system designers in the industry. We would examine their design strategies in the industry context using ethnographic methods and we would evaluate their designs in experimental situations using HDS as a tool. This would all feed into the extension of both HDS and the theory on which HDS is based.

The culmination of the design aiding approach could be the development of an intelligent design aiding tool. That is, a user would be able to actually modify a model in HDS to reflect his or her own design and content requirements. The design aiding component would then provide advice on the strengths and weaknesses of the modification and offer alternative design strategies as appropriate.

The second path we could take would be developing HDS as an instructional system. Under this approach the model systems provide a simulation environment and the design aiding system provides the feedback on performance. A trainee could be provided with design goals or with a model that has "bugs" in it. His or her task would be to either debug or to design to meet certain goals. The design aiding system would monitor performance and provide guidance and feedback as appropriate.

Writing Online Information: Expert Strategies

(Interviews with Professional Writers)

by

Thomas M. Duffy, Thomas Gomoll, Kathleen Gomoll,
James Palmer, and Ann Aaron

Communications Design Center
Carnegie Mellon University
Pittsburgh, PA. 15213

1 March 1988

This work was funded in part by the United States Army,
Human Engineering Laboratory, Aberdeen Proving Grounds, MD
under Contract # DAAA1-86-K-0019 AC85.
Contract monitor: Maureen Larkin.

The importance of high quality documentation for computer software and hardware is now fully recognized. The quality of the documentation is typically a part of any software or hardware review and some magazines even give annual awards for "best documentation". This is a dramatic change for the attitude of just seven or eight years ago when the documentation was considered as an after thought and manuals tended to be written by programmers and considered system documentation rather than a tool for the user.

Just as the industry has come to develop effective, or at least reasonable, documentation standards, we see a new documentation challenge on the horizon: online documentation. By online documentation we mean a program and database that is a resource for using an application program or some piece of hardware (Duffy and Langston, 1985). The goal for online documentation is job aiding. Thus regardless of whether it is a the hardcopy manual placed online or it is separately designed documentation, the focus is on providing procedural and reference information. We are not considering computer-based instructional or tutorial materials as online documents since they have the goal of teaching rather than job aiding.

There are several reasons why online documentation is becoming increasingly common. Partly there is simply a greater urgency in insuring that computer users have the necessary documentation available. With the tremendous growth in computer users, companies are finding it increasingly difficult to provide the necessary technical support. Furthermore, the increasing variety of competitive applications and computer systems makes it difficult to find a fellow user who can provide on the spot advice.

As 800 number hotlines disappear, and developer support comes only through a long distance call, if at all, the software and hardware developers must insure that users have documentation available. While hardcopy manuals could fill the necessary information role, they are not always available. With site licenses and networking it is common to find the number of users outstripping the number of manuals. More basically,

however, manuals are not attached to the equipment and thus get misplaced or are simply inconvenient to access.

Cost and efficiency is a second reason for the move to online documentation. Online documentation is easier and cheaper to store, to distribute, and to update. This is a general issue with technical documentation, and there are numerous efforts outside of the computer industry to provide computer based documentation delivery for a wide variety of technical equipment. We might expect hardcopy manuals to be absent from the industry's ideal world.

While we see an increase in online documentation and we can anticipate that online support will soon be very common, we see very little in the way of guidance on the preparation of the documentation. An informal survey of several writers in the computer industry indicate that standards for online help are rare. Additionally, a review of the literature shows very little work aimed at providing guidance to the online developer.

One line of research has compared the presentation of the same documentation online or in hardcopy. Invariably, the findings have been that performance is more efficient and more is learned with the hardcopy document (Shneiderman, 1987; Cohill and Williges, 1987). However, these effects can be attributed to factors. First, in virtually all the research, the user could not simultaneously view the online document and the application program it was supporting. Thus, there were greater demands on short terms memory when using the online document. Second, people read more slowly from a computer screen. In general this seems to be due to perceptual factors involved in screen vs print presentation (Rubens and Krull, 1987) but there is also some evidence that online screens are read more thoroughly (Duffy, et.al. 1987). While this research identifies issues that must be addressed in the presentation of online documentation, it does not provide guidance for the developer.

A second line of research has focused on perceptual and typographic issues in presenting text on the screen. These are general issues in designing a human computer interface, but they also tend to be the focus of discussion on online documentation (see, e.g., Brockman, 1986). This research has demonstrated

that online presentations require more white space, require more leading between lines, and careful attention to the selection of fonts (Rubens and Krull, 1985 ; Brockman, 1986). Bradford (1984) outlines the use of emphasis devices (blinking, reverse video, color) and issues in positioning information on the display screen. However, there is precious little information beyond these basic perceptual issues.

We know that readers do not feel as comfortable reading through and editing a document online as they do in hardcopy (Haas, 1987), but we do not know how to organize the information to reduce that effect. There is little guidance and even less data on how to organize information in the overall database or even on how to present information on the screen.

Shneiderman (1987) in his excellent book on the human computer interface describes online documentation as "making technical manuals available on the computer"(p. 374). His only cautions in doing this have to do with the limitations of screen size on the amount of information (both application software information and documentation) that can be seen at once and on the greater difficulty of reading from a screen display. Thus one might suppose that technological improvements (screen size and resolution) are the only impediment to simply transferring hardcopy manuals online. There seem to be no new design or writing requirements.

Many online documentation systems, in fact, do involve simply placing the hardcopy manual online. However, most researchers and writers now agree that as documentation moves from paper to the screen there should be a basic conceptual change. It is vital that developers and writers see online information as software and not as electronically stored pages (James, 1985; Price, 1981; Richier & Thompson, 1974). Brockman (1986) notes that we cannot naively switch from writing paper manuals to writing online manuals without some added care, attention, and without acquiring skill in new techniques"

It seems clear that writers will have to develop new skills to meet the demands of writing this online information. But what are these new skills? What are the new guidelines? Many potential design features for online

documentation have been described or demonstrated in a prototype (Borenstein, 1985; Brown, 1985; Rothenburg, 1979; Engle and Granda, 1975; Price, 1986; Relles, 1979; Shneiderman, 1987; Walker, 1986; Houghton, 1984). However, our concerns are more basic. We do not want to know what could be part of an online documentation system but rather we want to identify the necessary design requirements.

Shneiderman (1987) and Borenstein (1985) indicate that the quality of writing is an important determinant of the effectiveness of online help. Of course, writing quality is an issue in the usability of any text. The question is, what is high quality writing in online documentation. Perhaps the most common advice is that the online documentation must be written more concisely and fitted to the screen (Brockman, 1986; Walker, 1986; Price, 1986; Houghton, 1984)

The goal of this research is to probe expert opinion on the issues and principles in the development of online documentation. In particular, the goal was to identify the training requirements for document developers new to online documentation and to identify the critical design principles that distinguish online and hardcopy documentation and the basis for those principles. We hope that the identification of this expert judgement will provide some guidance for writers and will identify key research issues in evaluating documentation strategies.

Method

Participants

Nine professional writers, working in the computer industry, were interviewed. Preliminary screening of the writers established that they all had at least five years professional experience in writing computer documentation and had had primary responsibility for developing at least one online help system for a commercial software product. The participants came from 4 different hardware and software companies in the United States. Participation was voluntary.

Procedure

An initial phone conversation with the participants was used to establish their eligibility, based on experience, for inclusion in the study and their willingness to participate. A time when the formal interview could be conducted was established.

Seven of the interviews were conducted over the telephone and two were conducted in face to face meetings. All participants were informed that the interviews were being tape recorded but they were assured that the data from all of the participants was being pooled and no response would ever be associated with a particular individual or company. The interviews lasted approximately 1 hr.

Interviews

The interviews were semi structured. They began with two questions establishing number of years of experience as a writer and the amount and type of experience in developing online documentation.

The participant were then told that we did not want to know what they did in the past in developing online documentation. Rather, we asked them to consider the following scenario in discussing online documentation with us.

Assume the following situation: You have just been given the responsibility for creating the online documentation for [whatever product they last developed help for]. The technical manuals for the product have already been written. A team of writers have been assigned to you to prepare the online documentation. They are experienced in writing technical manuals and they are experienced with [the product being documented]. However, they do not have any experience in preparing online documentation.

Given this scenario, the participants were asked to discuss what orientation and training the writers should be given in preparation for the project. It was explained that we wanted them to discuss in detail what was different between preparing hardcopy and online documentation and what sort of training was required to prepare the writers for those differences. The interviewer probed for more details on each training issue the participant raised so as to be sure that the issue was fully explained or described.

The interviewer had a list of issues to be covered in the interview. If the participant did not address those issues in the initial discussion of the scenario,. the interview raised them as formal questions and probed for more details during the participants response. The list of questions are presented in Table 1.

Results and Discussion

The tape recording of the interviews was transcribed and the participants comments were segmented into topic statements. A topic statement was any statement that mentioned training or skill required of the writers, a computer system or application software characteristic that affected the design of the online documentation, a design characteristic of online documentation, or a characteristic of the users of online documentation.

Two of the interviews were then used to develop a list of core topics. That is, the topic segments in the interview were grouped into categories. Thus, for example, a segment that stated the audience for online documentation is more in need of immediate assistance and another statement that users of hardcopy documentation are more willingness to read through the text

would both be classified under the same core topic: Two raters worked together to develop the classification scheme or list of core topics. Reliability of the coding was then assessed by having the coders independently rate the segments in another interview transcription. There was an 82.5% agreement between raters in terms of the frequency with which each core topic was mentioned. Each researcher then coded half of the remaining interviews using the master list. When we completed coding all the interviews, our final coding agreement was 90.1%.

The list of core topics and the frequency with which each topic was mentioned is presented in Table 2. While the respondents referred to many similarities between online and hardcopy documentation the focus of the interview, and the emphasis in their comments, was on the new issues to be considered when writing online information. Eight of the core topics accounted for 46% of all of the comments made by respondents. These eight topics, in turn, reflect the importance of the constraints of the computer system and of the perceived goals of the users on the the design strategy. The eight most frequently mentioned core topics were:

1. Access mechanisms (9%). The experts stressed the importance of determining how information will be accessed. Clearly there are a wide variety of options available in designing access to a topic and navigation between and through topics in an online system that simply are not available in hardcopy: menus, keyword, hypertext;, automated indexes, etc. And there are numerous variations on each of these options. Thus this becomes an important training issue for writers new to online documentation. ut it is not just that there are more options available. The writers emphasized the importance of the speed of the access and navigation system, how intuitive it is to the user, and how appropriate the structure is to the user's needs.

System speed, navigation, and the "visualness" of **online information** were also major concerns in the process of designing access mechanisms for online documents.

2. Modularity or chunking information (7%). Eight of the nine experts (89%) stressed the importance of producing chunks of information that are self-

contained (each piece of information can stand alone). By using self-contained chunks of information, users can ask to see any particular topic without having read any other part of the online information. Each chunk, or module, includes a strong topic sentence so the information can be read out of context (no introduction). Online help systems that rely on a database structure for retrieving information (as opposed to a file of running text) generally include modular text or chunks.

3. Screen size as it affects the format of information (6%). All of the experts identified screen size as a major factor constraining the design of online documentation. They all felt that the writer must know the size of the screen their text will be displayed on. There were differences between the experts in terms of how screen size should affect the design of the documentation. These differences will be discussed in the next section. The majority, however, stated that writing and formatting information for the screen affects the usefulness of the information. When well done, this writing and formatting can make the information more concise, more visually pleasing, and more readable. It appeared that most of the writers found that the screen size of online documentation is more of a constraint than the page size is in hardcopy perhaps because of the more visual nature of online information.

4. Organization/structure (6%). Five of the nine writers emphasized the importance of structuring information for online use. For example, using tables, bulleted-lists, headings, etc. are common ways of physically organizing information online. They insisted that a screen demands physical structuring as well as conceptual structuring even more than the page does because of the "disorientation" that many users experience when using an online document. They also discussed parallel information structure for online information; for example, this means using the same organization for every command that is described (i.e., include the same parts: summary, example, problems, related topics).

5. Screen design/format (5%). Writers stressed how screen design should differ from page design. They mentioned differences in layout that are required by the screen. For example, because readers cannot tell where they are in an online text simply by looking at it, writers often use grid formats to

place information consistently on the screen (i.e., Page 1 of 2). Writers also need to consider what information will be displayed on every screen, such as the command line or selections for the user's next action.

6. Conciseness (4%). Five of the nine experts said that online information should be more concise than hardcopy information. They claimed that people do not want to read lengthy texts online. So, on a small screen, some suggested the main goal is to get the message across in as few words as possible. One possible strategy is to use tables and lists as replacements for straight text.

7. Levels/hierarchies (4%). This refers to the global organization of online information. Generally, online information is organized hierarchically with several different levels and maps provide an overall picture of the hierarchy. Writers felt that the optimum number of levels in an online hierarchy was fewer than four. They noted that there was no parallel navigation problem in a hardcopy hierarchy because the physical manual provides a "compass" with which users can orient themselves relative to other sections of the documentation (e.g. "Now I am in the front of the book. Now I am in the second chapter," and so on).

8. Type of user (4%). Most of the experts indicated that they thought the user of online help differed from the user of hardcopy documentation. The users approach online help in a different frame of mind--they want answers immediately and do not care to read lengthy texts. This, then, has implications for the content and the style of online information: information must contain consistent terminology, must be concise, must treat each topic as separate paragraphs with clear topic sentences. Some would argue that these are also concerns of the hardcopy documentation writer, but the experts felt that such concerns were magnified in an online presentation mode.

These eight most frequently mentioned topics indicate that there are significant differences between preparing online and hardcopy documentation. One could look at the design issues and suggest that there is nothing new; the guidelines are simply those for good technical writing. However, the comments of the experts suggest that what is new are the

constraints in the delivery system and in the goals of the user. The shift in these audience and presentation characteristics are what lead to a significant shift in emphasis in the design strategy for online help. The simple statement of the guidelines may not have changed, but the goals in applying the guidelines and hence the way in which the guidelines are applied has changed significantly.

Perhaps one of the most important considerations in these comments is the impact of the new (relative to hardcopy documentation) approaches to access and navigation. Familiarity with the options for access and navigation and an understanding of the consequences of each option on user performance is an important new learning requirement for writers new to online documentation. In addition, however, the nature of the access and navigation can have a significant impact on the structure of the online information.

Constraints on the design of Online Documentation

The delivery system and the nature and needs of the audience are two of the primary factors or constraints affecting every aspect of defining the content and structure of information. Two of these constraints -- screen size and the goal orientation of the users -- were among the most frequently mentioned topics in the interviews. However, there were a number of other constraints identified by the experts. In this section we will describe the constraints on the approach to design that were identified by the experts.

System Constraints

Writers consistently mentioned four major constraints caused by displaying information on a computer system:

1. Screen size. All of the writers surveyed agreed that they write or format information to fit the screen. However, the guidelines for the format and content seem to change dramatically as a function of screen size. Five of the experts had worked almost exclusively with small screen (9"-12") presentation while the other four were working with large screen (19"-21")

delivery. Those who write for small screens tended to view each screen as independent. They agreed that:

-Writing must be very concise so the most important information can fit on the screen. Longer explanations are left out -- only the most vital information is included.
-Topics should be limited to 1-3 screens, and each screen should be able to stand alone. In other words, one screen does not run over onto the next.

In contrast, the experts who write for large screens generally compared the screen to a page in a book. While they saw screen format as important, it was in terms of the general screen design. They worried much less about tailoring the information for specific screen size, but instead emphasized the importance of including the information the user will need. Thus their design strategy leads to online documents that are generally more complete (not as concise), with more examples and other support information, more like a book.

It is not that the large screen developers did not emphasize format. Rather, they focused on text based rather than screen based formatting. They saw format as very important to the usability of the text. Format should support the coherence of the text and thus should be based on the content rather than the screen. One navigation implication of this approach is that while users of the screen formatted text must page through the information, the text formatted material may be either paged or scrolled.

We asked two questions after the expert completed comments on screen size to probe further as to the importance of screen size in the design process. First, we asked how the design strategy would change if there was a large (or small) screen display. Second, we asked how they would design online documentation for a system which gave the user control of the size of the window.

Those experts who wrote for large screens maintained their same design philosophy in considering small screen applications. That is, while they thought the amount of information on a topic would have to be reduced,

they still proposed a text like presentation with text based formatting. In contrast, four of the five experts who write for small screen had trouble imagining writing for a different screen size. They were not sure what design principles would apply, and thus were suggesting that their design principles could not be generalized. In contrast, one small screen writer argued that the online document should always be presented on a small screen. Thus, even with a large screen system the online document would be presented in a small window following the same minimal text and screen based formatting principles.

There was unanimous agreement among the experts that variable window sizes presented a major design problem. In general it was argued that the window size should be controlled by the developer; the user should not be permitted to modify the size of the online documentation window.

2. **Access mechanisms**. The limitations and possibilities for accessing topics and navigating through topics were identified as major determinants of virtually all aspects of the design of the online documentation. It is also a major factor in user acceptance of the system. That is, the experts indicated that the designer must be particularly concerned in developing access and navigation systems that are easy and intuitive to use, prevent the user from getting lost, and are structured so as to reflect the kinds of information and the links between information needed by the user. The experts all agreed that the writer should have some control over the design of the access and navigation systems, though most of them did not have such control when they developed an online documentation system.

3. **Memory of the system**. Although it is a problem which is rapidly disappearing, some writers pointed out that they weren't able to produce full-blown help text because of memory limitations. When given unlimited memory, writers stated that online help should include everything that a hardcopy manual does. However, when memory is limited, writers are forced to prioritize information: most agreed that online help should always include procedures and task-oriented information.

4. **Lack of context**. The need to contextualize information was reported as a major issue in the design of online documentation. The user can only see what is on the screen and thus it is essential that the designer provide the user with a sense of the whole text on a topic, the placement of the topic in the overall documentation, and of related topics. Mechanisms like structure maps ("you are here"), previewing, and presentation of parallel and hierarchically related headings were seen as useful mechanisms for providing context.

User Constraints

1. **Quick Solution.** Most of the experts stressed that people who read online information have very immediate needs; they are usually experiencing a problem, and they need a quick solution. Hence, online information must be more concise than hardcopy information and more geared to solving problems. Writers must constantly be aware that their readers will be in a problem-solving state.

2. **Variability in User Sophistication**. This is a problem in the development of hardcopy documentation as well, but the experts reported one issue particular to the online delivery. The variability in audience sophistication affects not only what information they need but the technological features they must use to get at the information. Designers may be tempted to use very sophisticated access and navigation strategies that highlight the capabilities of the computing system. Sophisticated user will be impressed by the tools and may well find them useful for accessing and navigating. However, the less sophisticated user frequently does not have the conceptual basis to easily use the tool in a productive manner and may be put off and reject the system because of the number or complexity of the navigation and access options.

The Relationship Between Online and Hardcopy Documentation

At the end of the interview, we asked the experts to discuss how they thought hardcopy and online should be linked in the development process. We found widely differing attitudes on this issue.

Four of the experts felt that online information should be written after hardcopy information. Their rationale was that hardcopy is more extensive and contains many different structures--introductory material, procedures for completing tasks, examples, a glossary, a reference section, etc. They felt that the online document would be a subset of the hardcopy and thus could be easily extracted from these structures and pared down. These experts also tended to believe that that online information is frequently an after-thought: it is often "tacked-on" at the end of the documentation process.

Three of the experts proposed that online and hardcopy information should be written independently. They believed that the constraints in the delivery system and the users goals were sufficiently different that the online and hardcopy had to be considered as independent documents. They felt that online information should be written in a different style or tone, formatted differently, and accessed differently thus required an independent development process.

Finally, two of the experts argued that a single database should be developed and both hardcopy and online information should be derived from that database. Considerations of both online and hardcopy use should determine the structure and the content of the database. The rationale for this proposal tended to be based on the ease of maintaining the documentation. That is, using a common database reduces the effort involved in updating and also insures the consistency of online and hardcopy information. However, the argument also tended to include the views that the hardcopy manual should follow the modular style and format usually seen in online information and that the common database permits cross referencing between hardcopy and online documents.

Training Writers to Write Online Information

The initial scenario presented to the experts asked them to identify the training they felt writers new to the design of online documentation would require. The training issues and the percent of experts identifying each issue is presented in Table 2.

Table 2.
Training Requirements for new online documentation writers that were identified by the experts.

skill	% of writers who mentioned it
Ability to write concise, often stand-alone chunks of information that fit a pre-determined screen size	89
Conceptual understanding of databases and how to link different pieces of information	78
Ability to use graphics effectively	78
Understanding of the entire online help system the information will be displayed on, including how the information will be retrieved	78
Planning and organization skills	44
Programming skills	22
Strategies for usability (human factors) testing	22

We identified the new online documentation writers as being experienced technical manual writers. Given this context it is surprising that the most frequently mentioned training requirement was the ability to write short, telegraphic chunks of information. Eight of the nine experts identified this as a skill that should be part of a training program, underscoring the greater importance of efficiency in online, as opposed to hardcopy, presentation.

Even those individuals writing for large screen displays felt that this was an important skill in writing online documentation.

Seven of the nine experts identified knowledge of databases and of database linkages as a critical skill to be trained. This is really the skill of being able to understand the structure of a database so that segmentation and linking required for the users can be achieved within the database. Thus the training issue includes principles of access and navigation, the important core topic discussed earlier. Seven of the nine experts also identified the development of graphics skills as an important training requirement. In part this seems to be due to the increased need for efficient presentation, i.e., much technical and procedural information can be more efficiently presented in a table or graph. However, they also suggested that the computer medium is more visual and graphically-oriented than the paper presentation.

The remaining training requirements were for planning and testing skills and some skill in programming. The emphasis on planning and testing simply reflects the difficulty of organizing a segmented database to accommodate the constraints of the computer system and meet the needs of the user. Interestingly, only two experts suggested that the writers should have programming skills when preparing online documentation. And those two individuals emphasized that it was only enough skill so as to allow the writer to communicate effectively with the programming supporting the online documentation system.

Writers' Responsibilities

Finally, we asked the experts to identify what writers were responsible for in the online documentation projects they had experience with. They identified eight responsibilities as shown in Table 3. They all indicated that the writers are responsible for editing the documentation, however, only seven experts indicated that the writers were responsible for the initial writing. In the other two cases subject matter experts or system programmers prepared the documentation for the writers to edit.

Table 3.
Responsibilities of writers identified by experts.

Area of Responsibility	% Identifying it
Editing	100
Menu Design	89
Content of online information	78.
Screen design	78
Database links	67
Graphics	56
Naming conventions for commands	22
Error messages	22

The writers have a wide range of design responsibility. While they seldom have responsibility for naming commands (22%), they are almost always responsible for the design of the menu (88%) and the display of the documentation on the screen (78%). Importantly, they also frequently (67%) have responsibility for specifying the links in the database. Thus, the writers tend to have overall responsibility for the organization, structure, and display of the online documentation. The experts did not indicate specifically

indicate how much latitude the writers had in these areas. For example it is not entirely clear if they could request programming modifications. However, the comments in general suggested that the writers were structuring information within the given capabilities of the software capabilities.

It is interesting that error messages are still considered as part of the application software and thus are seldom (22%) the responsibility of the writer. Perhaps that is the reason error messages are the way they are.

Conclusions

The goal of this investigation was to assess the new skill and design requirements called for in the development of online documentation. By online documentation we mean a computer database that is separate from another application program but is designed to assist users in effectively using that program. We turned to experts, experienced in preparing both online and hardcopy documentation, for their judgements and experience regarding these issues. The data suggests that there are significant differences in both the skills required and the design strategy.

The new requirements and strategies derive from two factors. First writers of online documentation while having pretty much the same responsibilities as hardcopy writers but they have many new options within those responsibilities. That is, the writers of online documentation tend to be responsible for the identifying the content and structure of the menu system and the information, including graphics, presented on any menu topic. They also tend to be responsible for the structure of the database. While this is similar to the hardcopy documenters responsibility for the index and table of contents, the book plan, and the page format and content, there are many new options in the menu and database structure and linking.

The second factor, that leads to new training requirements and design strategies, is the constraints on the context in which the information is used. There are two sets of constraints that differ from the context of using hardcopy documentation: the users goals and the delivery system. Users are seen as more goal oriented and less tolerant of irrelevant information when

using online documentation. The delivery system constraints differ from the manual in that only the screen is present and thus the context surrounding the particular screen of information is not always clear. Additionally, and perhaps because of the lack of context, the screen size is seen as a major constraint on design strategy. Finally, the access mechanisms available to the designer will constrain the organization and design of the documentation.

Given these new constraints and the new options the writers identified three primary skills in which developers new to online documentation should receive training: writing concise and modular chunks of documentation; using graphics effectively; and, a conceptual understanding of databases including strategies for linking nodes.

The experts also identified several design strategies that either differ from hardcopy design or receive a different level of emphasis than in hardcopy design. The agreed upon design guidelines were:

1. Prepare modular documentation so that the presentation on any one topic can stand alone. Strategies for doing this include:

a. Either avoid using or define terms the user may not know.
b. Clearly label the goal in presenting the information.

2. Be concise. While almost trite in the world of technical writing, conciseness of the information is seen as more important in online than in hardcopy presentation. The important considerations in thinking about writing concise units of documentation are: the greater procedural goal orientation of the users goals, the impact of the small screen on the perceived amount of information being presented, and the effects of the lack of context on how easily the user gets lost.

3. Minimize the amount of time or effort the user must expend in finding information. Users tend to be in the midst of task on the application software when they enter the on-line documentation and thus they will be less tolerant of time delays. Also, perhaps because they can not see the entire database and can not easily skip around as they want to the way users of hardcopy documentation can see and leaf through the book, online users will

also rebel against unnecessary or confusing database linkages. Thus it is important that:

a. The menu and the structure of the database reflect the users needs and way of thinking about the topic.

b. The number of tasks (menu selections, keystrokes, etc.) the user must engage in to get to a topic are minimized.

c. Navigation aids like bookmarks, backtracking, and jumping to the main menu or out of the documentation are available.

4. The size of the window in which the online document is presented must be fixed -- it should not be under user control. All of the exports saw screen size as a major factor determining the design of the online documentation. Screen size has to be fixed in order to fix on the format and the amount of information to be presented.

The experts also disagreed on two major principles.

1. Should the formatting be screen based or text based? Individuals writing for large screens indicated that while the writing should be concise and modular, it must be complete and well written. They centered on the text based communication, viewing the screen more as a window. In contrast, writers for the small screen argued for screen formatted text, i.e., the text information was designed to fit on the screen, formatting focused on screen placement, and all screens should have the same basic design. If screen based formatting is used, the users must page from screen to screen. Scrolling would destroy the effectiveness of the formatting. Because the screen is the basis of the format any variation in screen size may result in a major revision of the formatting strategy. Thus, the design strategies of the screen based designers can be generalized far less than those for the text based formatters.

2. How should the development of the online and hardcopy documentation be linked? The experts were widely divided on this issue, advocating there very different strategies.

a. Hardcopy is written and the online is abstracted from it. The basis for this view is that the hardcopy is more extensive and is a more central focus in the development process. The online is a subset of the hardcopy and thus can be easily abstracted after the hardcopy is written.

b. Hardcopy and online are written independently. The basis for this view is that the content, organization, and format of a document is determined by the needs of the audience and the constraints of the delivery system. Since both of these factors are so different for online and hardcopy, they simply must be developed independently.

c. Hardcopy and online should be developed from a common database. There are two bases for this argument. First, from a management perspective, it is far easier to maintain a single database and when updates occur they will be consistently made in both online and hardcopy. Second, the modular style dictated by a database was seen as appropriate to both online and hardcopy delivery.

In sum, the design of online documentation is different from the design of hardcopy documentation. There are new training requirements and there are different constraints on the design strategy. The design guidelines arising from these differences do not appear, on the surface, to be very different from hardcopy guidelines. However, it is the details of how these guidelines must be applied, the extent to which they must be applied, and their criticality in determining the usability of the document that distinguished online and hardcopy document development.

If we are to develop an understanding of the writing and design process that can be usefully applied in real world situations, then it is essential that there be a two way flow of information between researchers and practitioners.. The data reported here is expert opinion based on practical experience. It identifies what appear to be critical variables in the real world experience and thus should provide direction for the research community. Systematic research evaluation of these guidelines is required to more fully establish the importance of these guidelines as well as to more precisely identify the critical variables and the basis for their effect.

Appendix A
Core Topics by Interview

Core Topics by Interview

The following tables show the emphasis that each expert placed on the core topics. The tables give the expert's online information experience with different hardware and software and a general statement of the expert's background. Each table includes:

1. The core topics mentioned during the interview.

2. The relative number of times each topic was mentioned during the interview (percentage of total topics mentioned).

The core topics are arranged by frequency.

Table 1: Master List of Core Topics

Topic	% of Total Topics Mentioned (Total = 524)	% of Participants Who Mentioned Topic (N = 9)
Access mechanisms	9.17	88.8
Modularity of online information	6.68	88.8
Screen size affects format of online information	6.30	88.8
Organization/structure of online information	5.53	77.7
Screen design, format of online information	5.34	88.8
Conciseness of online information	4.39	88.8
Levels, hierarchies, maps	4.20	77.7
Type of user (mood of user when using online information)	3.82	88.8
Writer's responsibilities in the software engineering process	3.63	77.7
Screen size affects writing	3.44	77.7
Writer needs to know how the online system works	3.44	77.7
Different writing style for online information	3.24	55.5
Online information is a different medium:Visual/graphic medium	2.67	77.7

Table 1: Master List of Core Topics

Topic	% of Total Topics Mentioned (Total = 524)	% of Participants Who Mentioned Topic (N = 9)
Online information has very little surrounding context	2.49	66.6
Scrolling through online information	2.49	77.7
Content of online information is different from hardcopy	2.29	55.5
Hypertext strategies	2.29	66.6
Immediacy of online information	2.10	55.5
Content of online information is same as hardcopy	1.91	55.5
Online information should be non-linear, stored in a database	1.91	44.4
Online information should be small chunks of text	1.72	33.3
Using cross references in online information	1.72	33.3
Memory/hardware constraints	1.72	66.6
Using windows/writing for a default window size	1.53	66.6
Never put the hardcopy manual online	1.53	66.6
Put the hardcopy manual online	1.15	22.2

Table 1: Master List of Core Topics

Topic	% of Total Topics Mentioned (Total = 524)	% of Participants Who Mentioned Topic (N = 9)
Online and hardcopy information should both be writtten at the same time	1.15	44.4
Use guidelines to write online information	0.95	22.2
Writing process differs for online information	0.95	33.3
Online information should be written and stored as one large text	0.95	22.2
Screen size does not affect writing	0.76	44.4
Hardcopy should be written first	0.76	44.4
Writing style is the same for online information	0.76	11.1
New skills required to write online information: planning/organization	0.76	44.4
New skill required to write online informaiton: usability testing	0.76	22.2
New skill required to write online information: graphics	0.57	33.3
New skill required to write online information: programming	0.57	22.2
New skill required to write online information: media	0.57	22.2
Use examples in online information	0.57	22.2

Table 1: Master List of Core Topics

Topic	% of Total Topics Mentioned (Total = 524)	% of Participants Who Mentioned Topic (N = 9)
Online information is easier to maintain	0.57	22.2
Writer's do not need to know how the online system works	0.38	11.1
Online information should be written first	0.38	22.2
Immediacy of hardcopy information	0.38	22.2
Use a template to create online information	0.38	22.2
Online writing process is the same as hardcopy process	0.38	22.2
Use color for online information	0.38	22.2
Screen size affects the usefulness of online information	0.19	11.1

References

Borenstein, Nathaniel S. *The Design and Evaluation of Online Help Systems.* PhD Dissertation, Pittsburgh, PA: Carnegie Mellon University, 1985.

Bradford, Annette N. "*Enhanced User Interface Through Computer Tutorials.*" Paper delivered at the IEEE Conference on Professional Communications, Atlanta, GA, October 1983.

Bradford, Annette N. "Conceptual Differences Between the Display Screen and the Printed Page." *Technical Communication,* Third Quarter 1984: pp. 13-16.

Brockman, R.J. *Writing Better Computer User Documentation.* New York, NY: John Wiley & Sons, Inc., 1986.

Brown, P.J. "Making Unix online documentation more effective." *Microprocessors and Microsystems,* 9(7), 1985, pp. 346-349.

Cohill, Andrew and Williges, Robert. "Retrieval of HELP Information for Novice Users of Interactive Computer Systems." *Human Factors,* 27(3), 1985, pp. 335-343.

Duffy, T., Trumble, J., Isenberg, T., Janik, C. and Rodgers, K. *Learning with Online and Hardcopy Tutorials.* Technical Report CDC-32 (ERIC # 284 562), Pittsburgh, PA, Communications Design Center, Carnegie Mellon.

Duffy, T. & Langston, D. *Online help: Design issues for authoring systems.* Technical Report CDC-18 (ERIC # ED-267-810), Pittsburgh, PA, Communications Design Center, Carnegie Mellon.

Engle, Stephen and Granda, Richard. *Guidelines fo. Man/Display Interfaces.* IBM Poughkeepsie Lab Technical Report (TR 00.2720), Poughkeepsie, NY, December 1975.

Haas, C. *Computers and the Writing Process: A Comparative Protocol Study.* Technical Report CDC-34 (ERIC # 281-219), Pittsburgh, PA, Communications Design Center, Carnegie Mellon.

Heines, Jesse M. *Screen Design Strategies for Computer-Assisted Instruction.* Bedford, MA: Digital Press, 1984.

Houghton, Raymond C. "Online Help Systems: A Conspectus." *Communications of the ACM*, 27(2), February 1984, pp. 126-133.

James, Geoffrey. *Document Databases: The New Publications Methodology.* New York, NY: Van Nostrand Reinhold, 1985.

Melded, Marilyn. *When People Use Computers: An Approach to Developing an Interface.* Englewood Cliffs, NJ: Prentice-Hall, 1981.

Price, Jonathan. *How to Write a Computer Manual: A Handbook of Software Documentation.* Menlo Park, CA: The Benjamin/Cummings Publishing Company, 1984.

Price, Jonathan. *Help! How to Create Online Help.* Apple Computer, Inc., Cupertino, CA: Alpha Draft, July 1986.

Price, L.A. "Using Offline Documentation Online." *ACM SIGSOC Bulletin*, 13(2-3), 1981, pp. 15-20.

Relles, Nathan. *The Design and Implementation of User-oriented Systems.* PhD Dissertation, Madison, WI: University of Wisconsin, 1979.

Richier, D. and Thompson, K. "The UNIX Time-Sharing System." *Communications of the ACM*, 17(7), July 1974, pp. 365-375.

Rothenburg, J. Online tutorials and documentation for the SIGMA message service. *National Computer Conference*, 1979, pp. 863-867.

Rubens, Phillip and Krull, Robert. "Application of Research on Document Design to Online Displays." *Technical Communication*, 32(4), 1985, pp. 29-34.

Rubens, Phillip. "Online Information, Traditional Page Design, and Reader Expectation." *IEEE Transactions on Professional Communications*, 29(4), 1986, pp. 75-80.

Schriver, Karen A. *Designing Computer Documentation: A Review of the Relevant Literature -- Hardcopy, Online, General Applications.* CDC TR-31. Pittsburgh, PA: Carnegie Mellon University, Communications Design Center, 1986.

Shneiderman, Ben. *Designing the User Interface: Strategies for Effective Human-Computer Interaction.* Reading, MA: Addison-Wesley Publishing Company, 1987.

Talbott, Stephen L. "A Writer's Reflections on Product Design." *ACM SIGDOC Asterisk*, 9(1), February 1983.

Walker, Jan H. "Symbolics Sage: A Documentation Support System." In *IEEE Spring CompCon '84 Proceedings*, Spring 1984, pp. 478-483.

Walker, Jan H. "Designing and Implementing Documentation Online: Issues, Strategies, Experience." In *CHI86 Tutorial: Computer and Human Interaction Conference*, Boston, MA, April 1986, pp. 114-132.

Theory and Methodology for Evaluating Online Help

developed by

Thomas M. Duffy, James Palmer, John Trumble

Communications Design Center
Carnegie Mellon University
Pittsburgh, PA 15213

1 March 1988

This work was funded in part by the United States Army
Human Engineering Laboratory, Aberdeen Proving Grounds, MD
under contract DAAA1-86-K-0019 AC85.
Contract monitor: Maureen Larkin.

Abstract

The goal of this paper is to present a strategy for evaluating online help systems that will provide feedback on the usability of a help system, either in development or in the field. By online help we mean a program with a database that is designed to provide assistance in using another program. Our focus is on job aiding information rather than tutorials and other instructional material. In addition to the normal psychometric qualities desirable for any evaluation tool, we want the evaluation system to be usable by experienced end users regardless of their expertise. We also want an evaluation that provides diagnostic information about a given help system and permits comparisons between help systems.

The approach we took was to develop rating scales for objectively assessing a wide range of features of a help system. Two different rating instruments are being developed: Help Design Evaluation and Help Content Evaluation. The Help Design Evaluation focuses on the access to information and the comprehensibility of information in the help system. It can be used without using the application program -- it does not depend on the content or structure of the application program. The Help Content Evaluation focuses on the accuracy and completeness of the information presented. Since the goal of a help system is to aid job performance, accuracy and completeness are in reference to the information a user needs to support the kinds of tasks he or she does in the application program.

An organizing principle in developing both evaluation instruments was a focus on minimizing the processing time and effort required in getting help. Thus the items in each rating system are organized into eight categories reflecting the eight tasks a person engages in when seeking and applying help: formulating the specific need for help, accessing the help system, selecting a topic, scanning the help text for relevant information, identifying the correct type of information, understanding the information presented, navigating to other or related topics, and, finally, applying the help information to the application. Within each of these categories there are questions about the presence or adequacy of features that will facilitate or hinder the performance of the task.

The report includes two versions of the Help Design Evaluation instrument. Future work will include completing the Help Content Evaluation System, developing an example database to assist in judgement based ratings, developing a database of ratings on different help systems, and assessing the reliability and validity of the instruments.

Computers and computer software are becoming common components of virtually all military jobs. They are used for storing and accessing personnel records, for performing a wide variety of office activities, in training programs, and in the operation and maintenance of materials. Obviously, the quality of the user interface of these computer systems is critical to the effectiveness of the systems. The interface includes the software interface, as well as the technical manuals and the online help supporting the software.

Standards exist for the design of the software interface (Shneiderman, 1986) and there are a wide variety of specifications for the design of technical manuals (Hatterick and Price, 1981). However, standards for the design or evaluation of online help systems have yet to be developed. The problem is more general then the lack of an Army or even a military standard. Attention to the design and to the evaluation of online help systems is so recent that little in the way of guidance exists anywhere.

Perhaps the clearest indication of the lack of attention to online help is the confusion over just what is meant by the term. In some sense every aspect of the software interface can be considered help. The placement and nature of the prompts, the availability and structure of menus, the placement of information, the use of signals to indicate that the program is doing something during wait times are all "helpful" components of the program. But even outside of the interface, we hear the terms "online help"; "online documentation"; "context sensitive help"; "error correction"; "tutorials"; etc. Sometimes these terms are used synonymously and sometimes they refer to distinct systems.

Defining Online Help

In this work we define online help as a program and database that is a resource for using an application program or some piece of hardware (Duffy and Langston, 1985). That is, it is a program that, at a minimum, permits access to and navigation through a database. The help program and database are functionally distinct from the application software and access to help is initiated by the user.

Online help is distinguished from error correction or error feedback in that error correction is typically a part of the application program, rather than a separate program and is system, rather than user, activated. Thus, online help is a computer based information system accessed by a soldier and designed to provide the soldier with assistance in completing a task within another computer program.

The goal for online documentation is job aiding; it is a source of reference information and procedural instructions relevant to specific tasks or jobs. We distinguish online help from tutorials or training on the basis of this job aiding vs instructional function. The user of online help is seen as a soldier in the middle of a job rather than a soldier learning about a job.

We do not distinguish between online documentation and online help. We will classify a system as online help as long as it is a functionally distinct program and database designed to aid job performance. If a technical manual that was designed to aid a soldier in using a computer system were placed online, we would classify that online manual as an online help system. We would hasten to add, however, that we would expect such an online help system to be ineffective: the design of an online help system is quite different from the design of a technical manual (Duffy, Gomoll, Gomoll, Palmer, Aaron, 1988).

We learned long ago from attempts to transfer manuals to microfilm and microfiche that the transfer of a document to a new delivery system can present substantial redesign requirements. Each delivery system imposes a particular set of design requirements and opens new design opportunities. The efforts by all three services to develop computer-delivered technical documentation indicates the complexity of the task of transferring hardcopy manuals to computer- based delivery. The design of online help is a segment of the overall goal of delivering technical documentation over the computer. It is distinguished from the other components in that it is designed to support the performance of tasks on the same computer system through which it is delivered

Our Goal

The goal of this paper is to present a strategy for evaluating online help systems that will provide feedback on the usability of a help system, either during development or in the field. That is, we are providing an evaluation system that will help to insure that help systems provide the soldier with ready access to job-specific reference and procedural information in a way that will support effective and efficient performance.

The intended users of this evaluation tool are DOD personnel or contractors involved in the design or evaluation of software interfaces. They may be part of a formal development effort in a software engineering environment or they may be engaged in a less formal, field-based evaluation of existing software. Because individuals involved in these design and evaluation efforts come from a variety of disciplines, we make few assumptions about their skills and knowledge. A user may be a technical writer, computer programmer, systems engineer, human factors specialist, or a widely experienced end user. However, we do assume that all of the users have wide experience with computer programs, that they are not computer novices.

If this tool is to meet our goals for this audience it must have the six features outlined in Table 1 on the following page. Most importantly, the system must provide comparative, diagnostic information. That is, not only must it provide information on the strengths and weaknesses of the help system but it must assess the relative importance of those strengths and weaknesses. If the intended audience is to use the system, it must also be efficient and must not require extensive expertise in any of the disciplines related to design. Finally, as with any assessment instrument, it must be valid and reliable.

Table 1

Goals for an Online Help Evaluation System

- **Efficient**. It is simply a fact of life that few people are willing to spend the time or money on a lengthy and detailed evaluation unless it is an evaluation of a feature that is obviously system-critical, i.e., life threatening. Thus, if we expect designers to use the evaluation tool, it must be inexpensive to apply and require minimal time.

- **Diagnostic**. An evaluation that simply states that the help system is "difficult" or "easy" to use is inadequate. The evaluation tool must identify the features of the help system that make it difficult or easy to use and indicate which features the developers should revise.

- **Comparative**. There must be standards by which we can compare help systems. We will never have the "perfect" help system. There will always be features we can criticize. We must know the relative importance of the criticisms -- how important is the deficiency, how much can it impact the usability of the system. The way to provide that data is to develop a database on a wide range of help systems, especially systems that have a history of heavy use.

- **Minimizes the need for discipline-specific expertise.** We have defined the audience as coming from a variety of backgrounds. If a tool is to be usable by all the members of that audience, it cannot call upon detailed technical skills in any one area. What expertise is demanded can be satisfied through the use of examples as a basis for comparison. We should emphasize, however, that while our goal is to avoid the need for discipline-specific expertise, this should not imply that anyone can go out and evaluate a help system. There will most certainly be a requirement for broad experience and expertise as an end user of the system being evaluated and of help systems in general.

- **Valid.** The evaluation system must have both face validity and predictive validity. Face validity is essential for user acceptance. Users simply will not use an evaluation tool unless it appears to them to be assessing relevant features of the help system. Because the primary goal in applying the evaluation tool is to ensure that help systems are usable for the intended end user, the tool also must be able to predict those systems (or system features) that will be more or less difficult to use.

- **Reliable.** The evaluation tool should yield the same relative score and diagnosis of a help system regardless of which help designer uses it.

Existing Strategies for Evaluating Online Help

We examined the literature on evaluating or guiding the design of help systems to see if an evaluation system already existed that could be easily adapted for our application. This included the literature on the design and evaluation of human computer interfaces in general, of technical manuals, and of online help systems. In this section, we review five strategies identified in that literature review. However, all five strategies as they have been implemented fail to meet all of our criteria.

The problem in developing an effective evaluation instrument is inherent in the help system itself. That is, a help system is designed to aid an individual in using an application program, so the size and the complexity of the help system are partly determined by the complexity of the application program. There are three characteristics of the application program that will affect the ease of using the help system: size, complexity, and the range of tasks for which it is used. First, a small program, one with only 20 commands, will require a far less complex help system than a large, 100 command system. Second, the help system for a single mode application program, like a simple word processor, generally will be less complex than the help system for a multimode system where the options available to the user depend on the particular program context. Finally, the help system will increase in complexity as the versatility of the application program increases. For a single task application program -- even if it is a complex task -- the designer can create a flow chart of the system and write task-specific instructions. There is obvious potential for designing intelligent help that would lead the person through the task. However, it would be very difficult to capture the user's needs and tasks in a multitask, open-ended system, like an operating system.

This relationship between the complexity of the application program and the complexity of the help system presents the primary obstacle in evaluating help systems. Clearly, we need a system that can factor out the influence of the complexity of the application program while providing diagnostic information on the usability of the help system. With this issue in mind, let us now review the alternative evaluation strategies.

User Testing

One of the most common approaches to evaluating the usability of technical manuals is user testing. In using this approach to evaluate online help systems , we would give the user a task to do in the application program and we would provide the help system as an information resource. Errors (reflected in actual performance as well as in "think aloud" statements) and performance time would be used to index effectiveness.

User testing provides very effective diagnostic information. Think aloud protocols provide a detailed record showing where the informati n is incomplete, difficult to understand, inaccurate, or misleading. Research findings suggest that the user testing yields more effective revision data than expert review. Although it provides superb diagnostic data to support revision, there are two major shortcomings, which we discuss below.

First, user testing is not efficient. It is a very time consuming and hence costly process for which developers are hesitant to invest the necessary resources. In spite of the evidence for the diagnostic value of user testing, the only significant movement toward testing has been in the computer industry -- and this has been very recent and focused mainly on end-user manuals. This application in the computer industry seems to be largely a function of the loss in sales many companies have suffered due to inadequate documentation. In areas where there is less of a market factor, there is little support for user testing. For example, even though DOD contracts for the development of technical manuals typically require user testing, the testing seldom occurs (Duffy, Post, and Smith, 1987).

Second, and more importantly, user testing does not provide comparative information. It is simply an empirical methodology for collecting error data; there is no basis for categorizing types of errors or identifying the relative importance of errors. If two systems are being compared, we have simply two list of errors with no means of comparison. Since the complexity of the help system is in part a function of the complexity of the application, we cannot even use the simple metric of number of errors to compare the two help systems.

Time is another important criterion of help systems that user testing measures. Even if performance were error free, a help system would be unacceptable if it took too long to get help. Indeed, one might suspect that the time required to get information is a major factor in the rejection of help systems. But how much time is too much? Of course, it will depend on the complexity of the help system which, in turn, is dependent on the complexity of the application. Therefore, we cannot establish a time criterion for getting information; we cannot even use knowledge of other systems as a base for establishing the criterion. Again, we are left with no basis for comparing two help systems.

An alternative user testing strategy might be to design tasks in which the user only works in the help system -- he or she never has to use the application program. However, if time is the measure, the same problems still apply. The complexity of the help system is a function of the complexity of the application program. Thus performance even within the help system will depend on the complexity of the application program.

Critical Incident Testing

This form of analysis is a variation on the user testing approach which has been used very effectively in identifying critical factors in performing a task. The basis is that, when redesigning a system or task, it is most efficient to focus on the critical factors affecting task performance. In using the approach to evaluate online help systems, a group of users would be given tasks to do using the application program and the help system. As they go about performing their tasks, they would be asked periodically to identify the most serious problem and the most helpful feature of the help system that they have experienced since last reporting. Their answers become the focus of the revision effort.

This approach better meets our criteria for an evaluation system, but it still has some shortcomings. It is more efficient than more traditional user testing since many users can be tested at once and the detailed think aloud reports do

not have to be analyzed. This efficiency is at the cost of a decreased level of detail provided.

The main problem with this approach is similar to the problem with user testing: the data are still very specific to the particular application. It is an empirical approach yielding system-specific information but not contributing to the growth of an evaluation database on design features or strategies. The complexity of the application program being supported is still a dominant influence on the outcome of critical incident testing.

Key Task Evaluation

This is a performance testing approach that has been used successfully to compare different programs for the same basic application, e.g., different word processors. Key to the technique is a detailed analysis of the tasks a person performs in the application program. Roberts (1986), for example, analyzed the tasks of word processing applications. The key tasks in word processing were defined in a matrix where the rows represented word processing actions and the columns represented the objects (text components) on which those actions occurred. The cells in the matrix represented basic and general word processing tasks.

Roberts next defined a methodology for testing a user's ability to perform a key task using a word processor. Given the task set and the testing methodology, any two word processors could be compared in terms of accuracy and time required to perform the key tasks. When a new word processor is developed it can be tested following the specified method and compared to the existing database on other word processors. There have been several evaluations since Roberts' initial work that have contributed to the word processing database.

This approach fits our needs. It is designed to provide comparative data, the goal that protocols and critical incident data could not meet. Furthermore, by evaluating performance on the specific tasks, one could generate diagnostic data. Finally, the ability to conduct group tests and to focus on performance time and errors means the system could be administered efficiently.

However, as discussed in the previous section, the use of a help system involves using the application program and the complexity of the help system depends on the complexity of the application program. Thus, straight forward performance testing using key tasks cannot provide a meaningful comparative database; time and errors on the help system will depend on the complexity of the application it is supporting.

We considered whether we could use the key task approach with a particular spectrum of application programs. For example, could we use the performance of key tasks to compare the help systems for word processing programs, much as Roberts compared word processors? However, it is clear that even this restriction does not permit the development of a comparative database of performance times. In Roberts' case, the key tasks are simple movement commands (delete, copy, etc.) applied to different size chunks of text. The difficulty or time required to accomplish these tasks should not be a function of the complexity or power of the program. Indeed, we might expect that these tasks would be easier to accomplish with a more powerful program.

However, the variation in complexity of the application program would, by necessity, affect the speed of finding information on those basic tasks. It will always be faster to find information in the help system when the application program consists of 20 commands than when it consists of 100 commands. Similarly a poorly designed application program will place greater demands on the help system: the word processor EMACS (Gossling, 19), with its idiosyncratic naming strategy and many alternative ways of doing the same thing, will require a more supportive help system than a more intuitive word processor. Thus the performance of the help system tasks will in large part be dependent on the system being helped.

Ratings

It is now commonplace to find computer magazines and evaluation groups rating online help and documentation as part of the review of a product. These are expert judgements either expressed as a simple commentary or made on a five or ten point scale ranging from very good to very bad. This is

the first evaluation strategy in our list that does not rely on performance testing and thus some of the problems associated with that testing do not exist. For example, ratings do permit the development of a comparative database: we can discuss the ratings for the help system for product A versus the ratings for the help system for product B. Additionally, ratings can be obtained very efficiently. This approach offers some potential in meeting our needs, although there are several problems with the ratings or the way we have seen them implemented.

The ratings are done by experts who have wide familiarity with application programs, help systems, and documentation. This experience is essential since the raters must have some sense of what is possible. However, it also presents a problem for obtaining valid evaluations. Hayes (1982) found that experts have considerable difficulty detecting potential user problems. In essence, the experts don't experience the problems because their expertise leads them to effective strategies for avoiding or easily overcoming the problems. We would expect that they are much more able to fill in missing information (both about the program as documented in the help system and about the use of the help system itself) and would not tend to notice incomplete information. This poses some questions as to whether expert ratings are a valid indicant of usability for the typical user.

A strategy for overcoming the detection problem is to provide explicit evaluation criteria for the expert to focus on. This includes identifying the features to be evaluated and objectively specifying the criteria to be used in each case. In essence, the rater is simply rating the presence or absence of explicitly identified features. This approach would certainly contribute to the reliability of ratings. Of course, the validity of the system will depend on the questions and criteria presented.

Unfortunately, none of the rating schemes provide detailed rating questions. There are typically only one or two questions on the help system or documentation (e.g., "Is it easy to use?" or "Is it complete") and never more than seven questions. While the approach of expert ratings has potential, the existing rating systems must be considered to be of questionable validity and

reliability. Furthermore, because the questioning tends to be very general, the approach has not provided diagnostic information.

Specifications and Guidelines

Military systems, including technical documentation, are always developed to contractual specifications. For technical manuals and for interface design, the specifications include detailed requirements for the content, organization, and layout of information. For example, Hattrick and Price (1981) report that there are well over 100 different specifications for the design of just one type of technical manual, the job performance aid. Those specifications can be very lengthy. In one case, the presentation of the specifications requires more than 40 pages and there are another two volumes providing guidance in how to meet the specifications.

Of course the Department of Defense is not alone in providing specifications for the design of documents. Guidebooks have been around for hundreds of years and are fundamental to writing. Guidelines represent a basic approach to aiding the design and development process. For example, Strunk and White (1979), presents basic guidelines for any type of writing, Hartley (1981) presents 80 rules for the design of instructional materials, and Smith and Mosier (1986) present thousands of guidelines for human computer interface design.

We reviewed the guidelines and specifications for the design of human computer interfaces, technical manuals, and online help systems. Our goal was to determine if the systems of design guides could be easily adapted to an evaluation system in which experts rate a help system on the basis of the presence or absence of design features. In conducting the review we were interested in both the specific guidelines as well as the organizational structure in which the guidelines are presented.

Our review did not lead to any existing system that could readily be adapted either in terms of the set of guidelines or in terms of their structure. There are a variety of sources that discuss relevant features of online help systems (e.g., Shneiderman, Brockman, Walker, 1987). However, these sources are

not comprehensive and what they do present tends to be a mixture of necessary features (e.g., factors related to screen design) and enhancing features (e.g., natural language or execution of the program activity through the help).

There are, of course, numerous guidelines for interface design in general. This includes the military standard recently developed by the Human Engineering Laboratory (1985). We turned to these guidelines and standards to see if we could simply abstract that organization (structure) for our help system evaluation instrument. The listing of guidelines tended to be exhaustive and very uneven. For example, the interface guidelines presented in Smith and Mosier (1986) are simply too extensive to be usable in an evaluation process. In addition, there is considerable unevenness in the relative importance and independence of items, so ratings could not simply be combined.

Most importantly, the organizational structures in the listings of guidelines did not reflect our goal for evaluating. Because we want to assess the usability of a help system, we are interested in design features as they relate to the performance of tasks within the system. We see an organization based on a task analysis of the use of the help system as the most relevant structure for evaluation. None of the sets of guidelines we reviewed met that criterion.

The most common structure was to organize by system components, e.g., screen design, keyboard requirements, error feedback, etc. This is a topic-oriented approach that simply does not focus attention on the proper issues. Smith and Mosier (1986), in contrast, do organize their guidelines by user tasks. However, because they are addressing interface design in general, their tasks simply are not appropriate for a help system evaluation. For example, three of their six categories are: data entry, data transmission, and data protection.

Finally, Marshall, Nelson, and Gardiner (1987) organize their presentation of over 100 interface design guidelines based on an analysis of cognitive processing requirements. The effects of a design feature on cognitive processing is a central issue, since the primary goal of designing a help system

is to minimize the cognitive demands placed on the user. However, the cognitive feature is not the appropriate system for organizing the presentation of guidelines. All of the cognitive issues are a part of the larger tasks a person must perform in a help system (or in any application). Thus the organization by cognitive principles must necessarily violate an organization based on user tasks. The user of such a system must understand cognitive structures to even begin the application, but even that knowledge would be inadequate for most applications. For example, what cognitive principles would one look under to find guidance on helping the user to locate relevant topics? Or, what cognitive principles would a designer look under to obtain guidance on the layout of a screen?

Summary

In summary, none of the existing evaluation systems are adequate to support our evaluation goals. Performance-based evaluations are very specific to the particular system being evaluated and are subject to the influence of the complexity of the application program. They do not provide a basis for comparing help system designs or for evaluating data independently of the complexity of the application program.

Expert rating systems tend to offer greater opportunity for meeting our evaluation goals. However, the rating systems that exist are either not very diagnostic or are very open ended, which makes their reliability questionable. Furthermore, there is some question that experts can accurately rate the difficulties nonexperts would have in using the system.

Providing the experts with explicit rating criteria is one strategy for overcoming the validity and reliability concerns as well as for enhancing their diagnostic capability. However, an examination of existing design criteria for user aiding systems failed to yield an appropriate set of design features to be included in the ratings or even a structure for presenting design features to be rated.

An Evaluation Strategy

None of the evaluation strategies reviewed above adequately meet our needs. However, by combining the strong features of each of the approaches, we feel we can develop an effective evaluation system. As we have already noted, any evaluation approach that relies on performance measures will not lead to the development of a comparative database. The complexity of the application -- and the consequent effects of that complexity on the use of the help system -- will significantly contaminate the evaluation of the help system's usability. For this reason, we developed a system that uses expert ratings of the help system.

Since the goal of the evaluation is to assess the usability of the help system, an analysis of the tasks that a user performs in seeking help is required. The major processing tasks, identified through an information processing tasks analysis, provide the framework or organizational structure for the ratings.

Lists of guidelines are the source of the features to be rated. As we noted in the review of guideline systems, there are thousands of guidelines for document design, interface design, and help system design. We selected design features that could be seen as facilitating the speed or ease of completing each of the tasks involved in obtaining help information. Thus, the guidelines were all interpreted in terms of their effect on information processing time or effort.

Of course there are still hundreds of relevant guidelines. We further reduced the number of guidelines by examining existing help systems to identify features that distinguished one from the other. That is, we did not want to include features that all help systems tend to have or that no help system has. Our review included existing, commercial help systems, as well as prototype help systems described in the literature.

In the sections below, we describe in detail the components of our evaluation strategy: the separation of usability factors, the task analysis that provides the structure for the system, the cognitive features used in selecting relevant

guidelines, and the specific features rated in each evaluation instrument. The structure and critical features of the discussion are presented in Table 2.

Table 2

Summary of Critical Features of the Help Evaluation System

User Goal:

Locate information to resolve an impasse or to perform a task more efficiently.

Evaluation Goal:

Assess the usability of the help system in terms of:

- Access
- Comprehensibility
- Accuracy
- Completeness

User Tasks:

- Formulate a goal of information needed
- Access help
- Select a help topic
- Scan the help for relevant information
- Select relevant content/representation
- Comprehend text
- Navigate to related topics
- Transfer the help information to the application

Guideline selection criteria:

Identify guidelines that minimize the amount of :

- Perceptual Processing (searching/noticing)
- Cognitive Processing (deciding)
- Motor Processing (acting)
- System Response Time (waiting in performing each information processing task).

Evaluation goal: usability

The goal of the evaluation is to describe the usability of a help system and thus we begin with the requirements for usability testing. The usability of any information resource can be described in terms of four components: access, accuracy, completeness, and comprehensibility. When a soldier turns to an information resource for assistance, he or she will find it usable to the extent that potentially relevant information can be easily accessed. Once the information is accessed, it will only be usable to the extent that it is easy to understand. Finally, the information will only be usable if it provides accurate guidance and all of the guidance the soldier requires to complete the task.

In our approach to assessing the usability of online help systems, we consider accuracy and completeness separately from access and comprehensibility. Accuracy and completeness refer directly to the degree to which the information provided in help is adequate to assist the user in completing tasks in the application program. These are features that must be evaluated in the context of using the application program. This does not necessarily mean that assessing accuracy and completeness requires user testing as described in the previous section. It can still be done through user ratings. However, it does require that the evaluator attempt to complete tasks in the application program based on the available help information, as a user normally would.

In contrast, access and comprehensibility are characteristics of the help system used independently of the application program. They address the issue of locating and understanding information regardless of whether or not the information is accurate or complete enough to support tasks in the application program. These issues involve the design or structure of the help system and they can be evaluated without using the application program.

This distinction between the usability factors in terms of whether or not they can be evaluated independently of the application program leads us to the development of two distinct evaluation tools. First, we have developed a

Design Evaluation Instrument that assesses the online help independently of the application program. It is a rating of the design features of the help system. Second, we have developed a Content Evaluation Instrument that assesses the ability of a user to complete tasks in the application program on the basis of the guidance in the help system. The focus in the content evaluation is on accuracy and completeness. The issue is, given the information, can the soldier complete the tasks in the application software?

User Tasks

What do users do with a help system? While it's obvious that the help system must be usable, we must be clear in answering the question, "Usable for what?" The answer begins with a consideration of the goals of the user -- the soldier's purpose in entering a help system. As described in the introduction, we view the goal of online help systems as job aiding. Therfore, our overall context of using a help system is a soldier who needs assistance in effectively using the application program to complete some task.

There are two reasons why the user may need assistance. First, it may be that he or she is looking for a more efficient way to do a task. That is, the soldier can perform the task in the application but is looking for shortcuts. Alternatively, the soldier may have encountered an impasse. He or she has the goal of performing a task in the application program but does not know what to do next. It may be that the soldier simply does not know the next step in the procedure or that he doesn't know any of the steps and needs complete information on the procedure.

In all of these cases, the soldier is searching for very specific task-oriented or "how to" information. The appropriate information may be a procedure, information on a step in the procedure, or reference information to aid the soldier in generating his or her own procedure.

Given this context, let us consider a "coarse grain" analysis of the tasks the soldier must accomplish in obtaining and using the help. By "coarse grain" analysis we mean an analysis of the major tasks involved in the effort. A

summary of the tasks and the primary features in the help system relevant to accomplishing the task is presented in Table 3.

Table 3

An information processing analysis of the primary tasks in using an online help system and the major help system features related to executing those tasks.

INFORMATION PROCESSING TASK	RELATED SYSTEM FEATURE
1. Formulating an information goal	Names of help topics
2. Accessing the help system	Application Interface
3. Selecting a potential topic	Menu/keyword
4. Locating the information within the topic	Format of the help text
5. Selecting the type of information needed	Categories of information on a topic
6. Understanding the information presented	Comprehensibility of the help text
7. Going to related topics	Navigation system and help on help
8. Applying the help information in the application program	Windowing and program links to the application.

When the soldier requires help, his or her first task is to formulate some goal that will guide the search for information. Basically, the user must generate names or labels for the information needed. This naming will be based on the understanding of the task to be performed. The relevant issue in supporting this goal formulation is the extent to which the help system represents the possible task names with which the user enters the system.

Given that a help information goal is formulated, the user must access the online help system.[1] The soldier must find the route from the application

[1] Note that "access" here is a very specific task in using help. It is getting into the help system. This should be contrasted with the usability goal of accessing the help

program to the online help and this will, in large measure, be a function of how help is identified in the application interface.

Once the soldier enters the online help system, he or she must identify and select a topic that may contain the relevant information. This selection task involves the use of menu and keyword systems.

After selecting a topic, the soldier will enter a help text on that topic. It is seldom the case that the help text will focus exclusively on the information relevant to the soldier's needs. Indeed, it is often the case that a user is uncertain whether the topic selected contains relevant information. Thus the first thing he or she must do in entering a help text is to scan it to determine where relevant information may be located. The format of the help text and the use of headings will be primary issues in that scanning.

If the soldier identifies potentially relevant text, he or she must read and comprehend it. The writing style and the graphic elements are important factors in this comprehension task.

While the text may be relevant to the general topic, the soldier must have specific information on the task. Thus, the specifity of the categories of information available on a topic will be vital to successfully getting the right type of information. Additionally the information must be accurate and complete.

Finally, when the soldier locates and understands the correct information, he or she must transfer that information from the help system to the application program. The linkage between application and help is the feature relevant to this task.

These eight tasks provide the primary structure for both our Design and Content evaluation systems, as these are the user's basic tasks in using help. The evaluation systems simply ask questions about the design and the

---

information described in the previous section. The goal of accessing the needed help information includes a number of tasks, one of which is accessing the help system.

content of the help system, relevant to the successful completion of each of the tasks. The design questions will address the structure of the help system: what kinds of information are presented and how it is presented. The content evaluation will address the adequacy of the content for completing the tasks.[2]

This structure of the evaluation permits us to compare different help systems. We can, for example, compare the ease of accessing help in Microsoft Word to the ease of accessing help in UNIX. Even though the applications are very different -- a word processor and an operating system -- access is still a basic task common to the use of both help systems.

Within this structure we could simply ask experts to rate how well the content and design of the help system support completion of each of the tasks. That is, ask them to rate how easy it is to access help or navigate through the system; to rate the ease with which the task relevant information can be selected, scanned, and comprehended; or to rate how accurate the information for the task is, etc. However, we are concerned that such open-ended ratings would have low reliability and validity. In essence, the rating would depend on what features the rater happened to focus on. We anticipate that we can increase validity and reliability and also increase the richness of the diagnosis by asking for ratings of very specific system features as they relate to a task.

Guideline Selection Criteria

Our goal is to specify design and content features that an individual will rate in evaluating the effectiveness of the help system. The effectiveness of the help system is defined as the amount of time and effort required of the user to resolve the information need that generated the help requirement in the first place. That is, we want to evaluate the degree to which the content and design of the help system allows the user to get and apply the information he or she

---

2 "Content" refers to the content of application specific information.. The content as it pertains to the structure of the help system itself, e.g., the command or name used for accessing help, is part of the design evaluation..

needs in minimum time and with minimum effort. More specifically, the evaluation system will assess the time and effort required in each of the help tasks outlined in the previous section.

How do we assess the effects of a design or content feature on time and effort? Time is the amount of time it takes the user to perform each task. But, as we discussed previously, the evaluation goals are not supported by performance testing -- the most obvious way to obtain time and effort data. As an alternative to performance testing, we will use a rational analysis to identify design and content features that we may expect to increase time or effort in performing each of the tasks. In particular, our approach is an adaptation of the Card, Moran, and Newell (1983) model human information processor developed as a part of their analysis of human computer interaction.

They postulate that an individual uses three processing systems in interacting with the environment: perceptual, cognitive, and motor. Most every action an individual takes requires a response or an activity within each system. That is, the user must register a stimulus, link the stimulus to a response, and execute the response. For example, pressing the return key in response to any prompt requires registering the prompt (perception), "notifying" the motor system to respond (cognition), and then pressing the the return key (motor action). The number of activities will be increased to the degree that complexity is added to the system. Thus, if there are several possible prompts but only a particular one requires a "return", the user must register the prompt that appears (perception), determine if it is the designated prompt (cognition), notify the motor system to respond if it is (cognition), and press the return key (motor). In addition to these processing tasks, time will also be a function of the degree to which the user must wait for the system to respond.

Following this approach, our evaluation goal may be reformulated to be an assessment of the degree to which the help system minimizes the number of perceptual, cognitive, and motor tasks that the user must perform and the amount of time spent waiting for the system to complete each information processing task (or "unit task" in the Card Moran and Newall, 1983, formulation). We want to ask about design features that we can identify

appriori as increasing the number of these processing tasks. In the long term, we will want to complete the task analysis that will permit a detailed accounting of the task and time requirements. However, at this early stage of development such a detailed analysis simply is not feasible. Instead, we will use a gross consideration of the relevance of a design feature to cognitive, motor, and perceptual processing and to waiting. That is, inclusion of a design feature in the rating system will be rationalized based on the hypothesized increase or decrease in the amount of processing or waiting.

How might a design increase the amount of processing? There are four possible ways: the minimum number of processing activities may be high, the demands for recall rather than recognition may be high, the short term memory demands may be high, or familiarity of concepts and terminology may be low. We will briefly discuss each of these parameters in the following paragraphs.

First, a design may vary in terms of the minimum number of processing tasks the user must perform in order to get and apply the necessary information even when he or she knows precisely where to go for the information and how to get there. This may happen by increasing the number of stimuli to process (e.g., the number of items on a menu a person must search through), the complexity of the decision required to determine a response (e.g., the degree to which the user must interpret the appropriateness of a menu item or the help information), the number of decisions required to determine the particular response (e. g., entering a number or letter for a help topic vs clicking on it or typing it), and the number of keystrokes required in a response (e.g., the availability of word completion).

In addition to increasing the minimal number of required actions, the design feature may increase the likelihood of actions beyond the minimum by forcing the user to recall rather than recognize responses. If information is not visible on the screen, the user must generate an input. If there is not a match between the user's input and the label the system uses for the information, then the user is left in a loop of cognitive and motor activities, generating alternative strategies for obtaining the needed information. This could happen in accessing help (guessing what the entry command is vs

having it visible), in the selection of topics (having only keyword available), and in most of the other tasks.

Overloading short term memory is an obvious way in which the number of processing tasks can be increased. Presenting too much information and requiring the user to store it in memory will increase the likelihood of error and the need to either correct the error (if an action is taken) or to reprocess the information and develop a chunking strategy. The short term memory demands may be increased by presenting too many steps in navigating through the system, by requiring the user to remember one screen of information (application screen or help information) while looking at another, or by making unreasonable reading comprehension demands.

Finally, the number of steps may be increased by failing to accommodate the knowledge base of the user. If terminology and concepts are not familiar to the user, there is a need to search short term memory or to seek external sources of information to interpret the presentation. This potential problem is most obvious in the specification of help topics. That is, to the extent the help topics fail to reflect the user's representation of the information needed, he or she will have to search long term memory for synonyms that will link the representation to a listed topic or will have to review the problem and develop a new specification of the information need. Obviously, the link to long term memory is also important in comprehending the information presented on any one help topic.

In summary, design features of online help systems were identified in terms of their expected effect on the number of processing activities the user would have to engage in because of the complexity of the demand, the long and short term memory requirements, or the failure to support recognition. The items in the survey thus represent design guidelines based on a hierarchical information processing analysis. At the top level are the major tasks the user must perform. At the second level are design features that affect the number of processing activities the user will likely have to engage in in order to complete the major task.

Generation of items for the Evaluation Survey

We reviewed help system, interface design, and technical manual design guidelines to identify items for inclusion in the Evaluation Surveys. Each guideline was classified in terms of the information processing task it was relevant to and in terms of how it impacted processing activity.

This procedure resulted in a very large list of guidelines. Inclusion of all of them would make it infeasible to actually use the evaluation system, so we attempted to restrict the features to be evaluated in three additional ways. First, we examined a reasonable sample (25) of help systems for commercial software to identify those design features that are distinguishable between systems. Second, we attempted to consolidate guidelines that were closely related in terms of design characteristics into a single guideline or evaluation statement. Finally, we only addressed design features that one could reasonably expect in most help systems. Thus, for example, we did not include guidelines that addressed natural language processing or other features relevant to intelligent help systems.

Design Evaluation and Content Evaluation Instruments are being developed based on these criteria. We are still drafting the Content Evaluation Instrument, but we have completed two alternative Design Evaluation Instruments. The alternatives address the same basic design features, but represent the features in different formats. Design Evaluation Survey A presents each design feature as an individual item and the user simply rates the presence, absence, or inapplicability of the feature in the help system under review. It is an all or nothing rating. Design Evaluation Survey B attempts to address the degree to which a design strategy is implemented. Related design features will be grouped in a single item and points are awarded awarded according to the degree to which the design features are implemented. These survey instruments are presented following this report. We are currently assessing the reliability and validity of each approach. In the following section we describe the basic design issues being evaluated in each survey.

The Help System Design Evaluation Surveys

The surveys consists of eight sections, each section representing one of the information processing tasks. The evaluation criteria for each task area are as follows.

Formulating the information goal

When a user first encounters the need for help, he or she must formulate a *goal statement for the help being sought* (specify just what kind of help is needed) and then generate a name or names that they anticipate will be the labels used to index that information in the help system.

The expertise of the user with computers in general and with the application program in particular will have a major effect on how the information need is formulated and what names are generated to guide the search. A novice computer user may only be able to describe the need in terms of the real world task he or she is trying to accomplish. That is, the appropriate command names and computer tasks (e.g., opening a window) are unknown.

A transfer user, someone familiar with related programs but not this particular one, may have an idea of the type of command or the type of computer task but not know the labeling used in the particular application.

Finally, expert users will generally be very specific in implementing a command or requesting a topic using the exact terminology of the system.

An effective help system must attempt to support these different naming strategies (or naming problems, as the case may be). Failure to represent the task/command information in the way the user does means that the user will have to engage in additional hypothesis generation and needless exploration of irrelevant help texts. Thus the questions in this section of the evaluation address the adequacy of the topic listing strategies. We do not assess the adequacy of any particular name; that is a content issue that will be addressed in the Help Content Evaluation instrument. Rather the Help Design Evaluation focuses on the systems for presenting information.

The first question asks about alternative menu systems. In addition to an alphabetical command help menu for experts, we would hope to find a menu that organizes the commands by computer tasks, and a menu that is organized by the real world tasks that the user wants to accomplish using the application program. It is desirable to organize these different representations into different help systems. If there is only one menu then we would hope that these different representations will be included in that one menu.

The second question asks about strategies for accommodating synonyms of the sort of labels a transfer user might generate. A keyword system that contains a range of synonyms is the most efficient way to address this problem. A keyword system is one in which the user can enter a term or phrase and the help system would search for help on it. Note that it is not just the availability of a keyword system that is important, but rather the availability of a keyword system with a rich synonym database. We see the keyword approach as most appropriate for meeting this goal because a menu based presentation of a wide range of synonyms and partially overlapping terminology would present the user with a very complex and confusing search task.

Accessing the help system

Once the user formulates the information goal, he or she must enter the help system to search for relevant help. In accessing help, the user must first determine that there is a help system and then determine the action required to enter it. He or she must then take that action and wait for the system to respond. Finally, the user may or may not enter the help system at a point maximally relevant to his or her information needs.

Consistent with this analysis we have identified five design features that may affect the processing activity required of the user or the system response time. The first design feature concerns the requirements placed on the user in order to invoke the help system. Minimal cognitive and motor effort is involved if the user can simply invoke the help system from anywhere in the application

program. Maximum effort is required when the user must leave the application program in order to enter the help system.

The second feature asks whether the system ever prompts the user to seek help. That is, if the user is issuing illegal commands, we expect an effective help system to prompt the user to go to the help system.

The third design feature has to do with the obviousness of the user action required to invoke help. The less obvious or intuitive the action, the more cognitive processing that is required and the greater the potential for incorrect responses. This design feature includes both the availability of a prompt for accessing help and the intuitiveness of the required response. An extreme example of this was that one of the systems we reviewed required the user to type CTRL -] to enter the help system --- and it was not prompted in the user interface.

The fourth feature relevant to access is the time it takes to enter the help system once it is invoked. This is simply a matter of system response time. Time is not as critical a factor here as it is in navigating through the help system because the user has just encountered the problem and is probably still analyzing it. It is a task switching time and some delay is tolerable. At the intolerable end is one system we reviewed where it took 30 seconds to enter help after implementing the help command. (And we have been told of a system where it takes two minutes to enter help.)

The final access design feature is context sensitivity. The question is whether or not the help system places the user in help relevant to the problem or to the characteristics of the task. This may mean that the user is placed in a help menu system appropriate to his or her level of experience or past history or it may mean that the user is placed in a submenu of commands relevant to the particular context in which the problem was encountered.

Selecting help topics

Once in help, the user's next major activity is to select something to look at, to locate relevant information. In this case we are first concerned with

locating any portion of the needed information. This task involves searching through a menu system or typing in key words in an attempt to find terms that seem to be relevant to the information needs.

The focus of the evaluation is on the structure and responsiveness of the system for selecting an item on which to get information. The particular names used in the menus are taken as a given in the Design Evaluation tool. The appropriateness and completeness of the menu entries will be addressed in the Content Evaluation.

The goal in supporting item selection is to provide as much well organized information as possible on a single screen so that the user gets maximum information for any given response. This is contrary to a common recommendation that menus should reflect a balance of breadth and depth (see, for example, MacGregor, Lee, and Lam, in press) or, more specifically, that there should be eight to sixteen items on a menu. The recommendation for a balance of breadth and depth is based on an analysis of the average number of entries a person has to scan in getting to a desired topic. For example, in a menu of 40 items a person would, on the average, scan 20 items before selecting. Balancing of breadth and depth would result in a 5-item top menu with a 8-item submenu for each top level entry. Here a person would only have to scan, on the average, 2.5 items plus 4 items for a total of 6.5 items.

Consistent with our information processing analysis, this latter arrangement would reduce the number of perceptual judgements. However, there are two problems with the depth/breadth balance. First, most errors in accessing information in a hierarchical menu system occur at the top level. Users select the wrong path right from the beginning and then have to back track. Second, the approach ignores the potential for categorizing and labeling the categories in the main menu. There is no reason that would prohibit the items on the 5 item menu from serving as category labels and the basis of organizing the 40 item menu. If screen size prohibits presenting all of the commands on one screen then pop up windows should be provided as part of the main (and only), menu. We would argue for this single menu approach even when the application program involves several modes with subsets of

commands relevant to each mode. If space permits, all commands should be placed on one menu and the mode names would become a means of organizing the commands.

The first evaluation question asks about the number of items on the top level menu. Consistent with the orientation expressed above, having a large number of topics on a menu is rated as good. However, this information goal must be balanced with the quality of the format to support scanning. The ideal is to present as many items on the screen as possible and still maintain a quality format. Since most monitors are thirteen inches, we have formalized this rating into a statement that the number of menu items on the main screen should be between 15 and 50. Having over 50 items is rated lower with the assumption of the small monitor. The poorest menu rating is for displays containing fewer than 15 items (assuming that the command set is larger than that).

Some programs we reviewed place a large number of items on a single menu but then require the user to scroll the items through a window that will only reveal a small subset at once. This seems to us to force the user to scan all items (since he or she does not know what is not being presented unless some other means of organization, like an alphabetization, is used in conjunction with this single menu), and thus presents the greatest cognitive demands of all.

A third question asks about the number of items on the submenus. We want to emphasize that submenus should not be used to display only a few items, since then the submenu items could be part of the main menu. Thus, systems are downgraded if they contain 7 items or fewer.

A fourth selection question asks about the link of a submenu to the rest of a help system. Basically, the idea is that a submenu should not stand off on its own. It either should be integrated into the main menu (e.g., as pop up menus) or integrated into the presentation of the help text (for example, through a "see also" listing or through a hypertext-like methodology where items in the text may be selected for more information).

A fifth question emphasizes the need to present the menu options in an organized way to help minimize the number of entries the user must scan. The most effective organization is one in which the items are grouped based on shared relevance, and the groups are labelled. We cannot judge the adequacy of the relevancy grouping since this would involve an analysis of the functioning of the application program and that is part of the Content Evaluation. We can only judge that there is a grouping and it appears to be based on placing related commands together.

The final question asks about the selection methodology. A selection should minimize motor activity -- and minimize the user's requirement to do more than simply select the item one has judged as relevant. The highest rated approach is either mouse selection or typing a number associated with the selection. Using the mouse is the most direct cognitive task, but mouse movement is less efficient then typing a number. Less desirable because of the motor effort involved, is to move the cursor to a selection by using the arrow (or other) keys.

Scanning help: Format of the help text

The format of the help text should facilitate scanning for relevant information. A format and organization that is applied consistently across help topics is essential to facilitating the user's search. Within a help topic, the information should not be so jammed on the page that the user gets lost in trying to find relevant information, and should not be presented in so small a point size (or in such detail, for graphic material) that the user must strain to perceive the information. The information should be organized functionally and labelled so the user does not have to read everything in order to find the information that is most relevant.

We rate a system higher to the extent that a format is applied consistently across help topics. For this item we don't assess how "good" the organization is, just how consistently it is applied. A user can learn to get around in a help text as long as there are consistent rules for layout, even if those rules do not match the user's needs.

We rate a system lower if a user scrolls rather than pages through the help text on a topic. Users tend to lose their sense of the whole document and their place in it when the document is presented on a screen. We see paging, with distinct labelling of pages, as important to helping the user find relevant information within the help text.

There are a number of questions related to the structuring of information. Lists of information should be presented in a list format and not as running text (the items should also have a parallel structure as we indicated in the evaluation of comprehensibility). Subtopics within the text should be clearly separated and labels should be used so that the user can scan for the appropriate topic.

We also rate the density of information on a screen. Users become frustrated and see the page as too cluttered on a screen at much lower density levels than they do with a hardcopy document. Rubin and Krull (1985) report that a density of 50% is the maximum screen density to employ. However, that is actual density and a screen always appears much more dense than it actually is. Since one of the goals in developing this Design Evaluation Survey is that it be easy to use, we have chosen to have evaluators rate the apparent density rather than actually calculated density.

One question addresses the font used in presenting text. We do not base the ratings on size or font selection for the typeface for two reasons. First, the acceptability of a particular font or size will depend on the characteristic of the particular monitor on which it is displayed. Second, a rater could not be expected to be able to identify all fonts or letter sizes. Therefore we simply ask the rater to indicate whether or not the typeface is easy to read.

Finally, there is a rating on whether or not the text information is presented in a mixture of upper and lower case letters. We expect that virtually all systems will receive a positive rating on this criterion. The user rebellion against the all upper case presentation of computer information has been well publicized and we seldom see a violation of this very basic legibility requirement. We include this item, even though it is not expected to have

much discrimination power, simply because of the dramatic negative effect violation of the principle has on legibility.

Obtaining the needed information: Content

When a user reads a help text, the content should be relevant to his or information needs. Once again, we can only judge the accuracy and completeness of the information through an examination of the application program -- and that is part of the Content Evaluation. In the Design Evaluation our concern is the extent to which the help system contains categories of information and representations of the information that are likely to be relevant to the users needs.

Thus, one of the questions asks specifically about the classifications of information on a help topic. The system is downgraded to the extent that the following categories of information are relevant but not included in the help information: syntax, function, related commands, possible applications, concrete examples of how to use the command, bugs, warnings, and tutorials.

Two additional questions address the extent to which the help information can be tailored to the user's needs. The first asks about levels of explanation: is there quick reference as well as full explanation, or are there novice and expert help presentations? The second question asks whether the help system aids the user in zeroing in in his or her needs either by providing the semantic context for a topic (e.g., previewing strategies) or through a dialogue probing for the appropriate context. This item is most relevant to systems with large help data bases.

The final question asks about the degree to which the help presentation is task oriented. This question reflects the importance we place on job aiding as the primary goal of a help system.

Comprehending the help text

While appropriate categories of information may be presented, the help system is not efficient if the user cannot understand the text. Indeed, the use

of "computerese," a special language common (though not unique) to programmers, has been a major complaint by users attempting to comprehend both hardcopy and online text. In this section of the evaluation we present nine questions about the incorporation of basic principles of plain language in the text. The questions are at the word level (e.g., the use of noun strings, personal pronouns, and technical terms) and the sentence level (e.g., the use of passive sentences and complex syntax). There are also questions addressing the use of graphics to summarize specification data and to show relationships.

Most of these items call for judgements or technical knowledge of English grammar. For example, one question asks if the sentence structures are "overly complex" while another asks if lists have "parallel structure." The goals in the design of this survey are to have an objective rating system that does not require technical knowledge. In order to meet these goals, we are developing examples of text representing each alternative answer to a technical or judgement-based question. These examples are not yet completed.

Navigating through the help system

Once the user makes a selection from the menu, he or she will go to that entry. After entering that particular help topic, the user may want to go to additional topics for at least three possible reasons: the topic did not have the necessary information and thus the user is still searching for the required help; the topic contained portions of the needed information, but the user needs to go elsewhere to get all that is required; the user is interested in getting information on related topics or some other, independent, topic.

The movement from this initial topic to other topics is the issue of navigation. Once again we want to consider design features that we feel will affect the cognitive, perceptual, or motor task requirements or which will affect system response time. In order to navigate, the user must understand and remember the principles of navigation. Even the most well defined navigational system will be ineffective if the user does not know or remember the principles of operation. This includes understanding the

structure of the help network, understanding the kinds of help information to be found under different sorts of topics, and knowing the command set for getting around.

Given that the user understands the design and knows the command set, it is then a matter of the efficacy of that design that facilitates moving around in general as well as moving between related topics. That is, the linkage between help nodes and the principles of movement should be designed to facilitate general browsing and should also facilitate the user obtaining all relevant and related information on a given task or topic.

Regarding this latter point we must emphasize again that we are not judging the appropriateness of the information presented, i.e., we are not judging whether the information the system judges as related is indeed related or is all of the related information that should be linked. Rather, our concern is, first, whether or not related topics are even considered and, second, the adequacy of the design for facilitating movement to the related topics.

There are six design characteristics relevant to the evaluation of these aspects of navigating. First, there is the question of whether the help system provides any help on using the help system. That is, when the user enters help, is there a prompt that tells him or her how to get information on the design of the help system? If there is help on help, does it provide structural information on nodes and the kinds of information given in nodes? If it is a complex help system (either because of the design or because of the complexity of the application program), is a functional model provided either directly or via analogy to aid the reader in understanding and using the system? Again, if it is a complex help system, are examples provided to aid the reader in understanding how to use it?

The next question addresses how easy it is to move between topics. For very large programs or for naive users, the use of submenus will probably reduce the cognitive effort in determining related topics and the perceptual effort in locating them. However, it is important that the user be able to easily move between menus and between help information and menus. The most demanding system is one that requires the user to always return to a main

menu before accessing a new help topic. It requires extra motor and extra perceptual effort.

The third question simply asks about the time it takes the system to respond to help requests. System time is more important here than in the initial entry into help for two reasons. The user is typically making multiple help requests and therefore any delay in the system is magnified. Also, the user has already formulated the problem and is in hot pursuit of information and thus delays will be frustrating. In our reviews we generally found the response time to be less than two seconds.

The final three design features of navigation are all captured in one question. The three features all have to do with aiding the individual in effective navigation. A system is highly rated if it has all three and is rated lower for each feature missing. The first of these features is some mechanism for recovering from getting lost. Complex data bases and navigating strategies will be the most powerful but they also offer the greatest possibility for getting hopelessly lost in the network. Some means for recovering is required if the user simply doesn't know how to get back to the beginning.

The second navigation aid is a method for bookmarking. If the user is in a help database and is searching for information, it is often the case that the he or she will want to go off and "check something out." After a temporary excursion the user should be able to easily return to where the branching began.

The final aid is the consistent availability of prompts on the commands used in navigating. The navigating commands should be on the screen at all times while the user is in help or else a prompt for a pop up window containing them should always be available.

Transferring help to the application

Once the user has the necessary information he or she must return to the application program and apply it. This linking of the help system to the application has been one of the biggest problems in the design of help

systems. Before windowing became common, either the help or the application could be displayed, but not both. A user would have to remember the problem in the application when he or she entered help and remember the help information when returning to the application. This taxing of short term memory led to much poorer performance in comparisons of online and hardcopy help systems.

While some of the problems in linking help and the application can be overcome through the use of windowing, there are still numerous programs that require the user to leave help to go back to the application, still placing a heavy burden on short term memory.

There are four questions that address the linking of help and the application. Three of them focus on the short term memory issues in transferring the help. One asks whether the application can be viewed while using help, another asks whether help information can be transferred to the application through cut and paste, and the third question asks if the user can work in the application while viewing help.

The final question on linking asks about the similarity of the structure of the information in help and in the application program. Basically, the layout of information in the help system should mimic the layout of information in the application program unless that layout would violate other design principles judged to be more important. Thus, if commands are organized in a menu bar in the application program, those menu bar items should serve as the category labels for presenting the commands in the help system. It may be that the organization of the commands under the application menu bar does not match the user's needs very well (i.e., it is a poor organization). Nonetheless, if all of the commands can be displayed on the help menu, this organization in the application should be used. The help system, in this way, will help the user learn the organizational structure in the program. The organizational structure should be violated only if it is poorly done and all of the commands cannot be displayed in the top level help menu.

Conclusions and Plans

The Online Help Evaluation System is designed to provide a methodology for diagnosing and comparing online help systems that is reliable and valid, and that can be used by technical writers, programmers, or system developers. Our approach has been to develop a rating system that directs a rater's attention to very specific features of the help system. The features are chosen on the basis of their relevance to the information processing demands a user encounters in obtaining and applying help information. They are organized in eight categories reflecting the eight major information processing tasks a user must engage in in the process of getting help.

Because of the influence of the application program on the usability of the help system, two separate rating systems are needed: a Help Design Evaluation that addresses the help system design independently of the application, and a Help Content Evaluation that asks about the accuracy and completeness of the the help information for supporting user information needs.

While we have thoroughly addressed the rationale for the Help Design System, we have just begun the development of the evaluation tools. We have completed two versions of the Help Design Evaluation instrument and have the beginnings of a Help Content Evaluation instrument.

Our next step will be to apply the two Help Design Evaluation instruments to a wide variety of help systems for commercially available software. We will have two raters evaluate each help system so that we can assess reliability. We are still developing strategies for assessing validity. In part it will come from the raters' feedback as to whether they felt the evaluation tools captured the problems with the help systems and ordered the systems (in terms of quality) in accordance with their subjective ordering.

If the expert judgements seem to be a reasonable basis for judgement, we may formalize that process and assess validity of the evaluation tool on a new set of programs by comparing independent expert ranking to the ranking based

on the ratings assigned using the evaluation tool. Alternatively, we could conduct user tests of the help systems to see if people have problems where the ratings suggest they will have problems (e.g., accessing help, comprehending help, etc.) and find the going easy where the ratings predict the system is well designed.

The ratings of existing help systems, assuming reliability and validity, will also provide a database that can be used in comparing other help systems.

We will begin the reliability and validity analyses using both Help Design Evaluation instruments, but we intend to discard one of them as soon as feedback indicates a strong preference for one or the other approach.

While the Help Design Evaluation Instruments are considered working drafts, they have already been through several revisions and we do not expect to see significant changes in the content. However, we still must generate examples to support the ratings. That is, two goals in the design of the system are to maintain objective ratings and to avoid the requirement for expertise in any specific discipline (though a wide range of experience with help systems is considered important to applying the ratings). Many of the questions do require some expertise in document design and many questions call for the rating of the adequacy or ease of use of some part of the help system. We want to provide examples that will define unfamiliar concepts and that will anchor the extremes of the judgement ratings.

We see the examples as being presented as a separate text, in parallel to the Design Evaluation instrument. That way, the examples will be available as an aid when needed, but will not hinder the evaluator when not needed. We anticipate that the example base and the reliability data will be available by May of 1988.

The Design

of

Online Help Systems

A Design Evaluation Instrument (A)

developed by

Thomas M. Duffy
Communications Design Center
Carnegie Mellon University
Pittsburgh, PA 15213

1 March 1988

This work was funded in part by the United States Army,
Human Engineering Laboratory, Aberdeen Proving Grounds, MD
under contract # DAAA1-86-K-0019.
Contract monitor: Maureen Larkin.

Overview

This survey is for the evaluation of the quality of the design of any online help system. (An online help system is a computer program that provides job aiding information on the use of other application software.) The help information is presumed to be accurate and complete.

Eight design features of the help system are evaluated: support for problem representations, access to help, the menu structure, the format of the text, the kind of content presented in a help text, the comprehensibility of the content, navigation through the help system, and the linkage between the help system and the application software. The rating of each of these features is based on the amount of cognitive, perceptual, and motor effort that segment of the design imposes on the user.

The analysis yields a score for each component as well as an overall system score. The component score is simply the ratio of the number of positive characteristics of that component of the help system to the total number of applicable characteristics for that component. The System Score is the average of the component scores. Thus, all components are equally weighted in calculating the system score, even though the number of questions varies across components.

Instructions

1. Complete the Evaluation Background on the next page.

2. Score the help system on each question in each of the eight Evaluation Categories. The following is guidance for selecting answers.

Select	If
"yes"	The statement is true or generally true about the help system.
"no"	The help system does not have the feature but could. For example, most help systems will not have tutorials linked to the help text (question X under content) but such a linking of tutorials would almost always be possible and useful.
"N/A"	The help system does not have that feature either because it simple is not possible given the application software (e.g., submenus when the command set is only 15) or given the lack of a feature already identified by an earlier question (e.g., a question about the quality of graphics when an earlier question already established there are not any graphics.

3. Total the "yes" and "no" answers in the space provided on each of the Evaluation Category pages.

4. Copy the scores in the appropriate space on the Evaluation Summary.

5. Total each column on the Evaluation Summary.

6. Calculate Component Scores and System Score for the software. A well designed component well have a score near 1.0 and a well designed system will have a score approaching 100. Use the System Score and the Component Scores to compare help systems.

Evaluation background

Software name_____ Version Number_____
Manufacturer_____
System tested on_____

Evaluator_____ Date_____

Evaluation Summary

	A. Number of Yes Answers	B. Number of No Answers	C. Total Number of Questions	D. A + B (Total Relevant Questions)	E. A/D (Component Score)
Represent.	_____	_____	2	_____	_____
Access	_____	_____	7	_____	_____
Menus	_____	_____	11	_____	_____
Navigation	_____	_____	18	_____	_____
Content	_____	_____	11	_____	_____
Comprehens.	_____	_____	10	_____	_____
Format	_____	_____	9	_____	_____
Linkage	_____	_____	4	_____	_____
		TOTAL	**70**	_____	_____

SYSTEM SCORE INSTRUCTIONS: **a).** Sum column E; **b.)** divide by 7; **c).** multiply by 100.

The score may range between 100 (best system) and 0 (worst system).

SYSTEM SCORE: _____

CATEGORY I: PROBLEM REPRESENTATION

Does the help system support different naming for tasks and commands?

REPRESENTATION TOTAL: Yes _____ No _____

1. Yes __ No__ N/A__ Is there a keyword system with synonyms for commands and tasks?

2. Yes __ No__ N/A__ Are there menu entries or an index representing real world tasks?

3. Yes __ No__ N/A__ Are there menu entries or an index representing computer tasks (as compared to computer commands)?

CATEGORY II: ACCESS

How easy is it to access the help system?

ACCESS TOTAL: Yes _____ No _____

1. Yes __ No__ N/A__ Can you access help from anywhere in the program?

2. Yes __ No__ N/A__ Can you access help without exiting the program?

3. Yes __ No__ N/A__ Is a prompt telling you how to access help always present?

4. Yes __ No__ N/A__ Is the command for accessing help (or the menu name) intuitively obvious (e.g., control Help) or common (e.g., F1 on the IBM computers)?

5. Yes __ No__ N/A__ Does it take less than 4 seconds to access the help system?

6. Yes __ No__ N/A__ When you first enter help are you placed in a subset of help relevant to your current place in the application program?

7. Yes __ No__ N/A__ Does the help prompt you seek help when you commit errors?

CATEGORY III: MENUS

How easy is it to locate information in the menus?

MENU TOTAL: Yes _____ No _____

1. Yes __ No__ N/A__ Is there a menu of help topics?

2. Yes __ No__ N/A__ Does the top level menu contain all of the help topics and subtopics? If not, is the screen reasonably full of topics (e.g., 40 or more topics on the IBM screen and 25 or more topics on the Mac screen)?

3. Yes __ No__ N/A__ Is there an average of 7 to 15 topics on the menus below the main menu?

4. Yes __ No__ N/A__ Are all items on a menu visible at once?

5. Yes __ No__ N/A__ Can you examine submenus without leaving the main menu?

6. Yes __ No__ N/A__ If there are submenus, is the submenu a part of the help information presented on a topic (either embedded in the text through hypertext or as a menu at the end of the help information)?

7. Yes __ No__ N/A__ Are items on the menu organized into groups based on shared relevance?

8. Yes __ No__ N/A__ Is there a heading for each group of items on the help menu?

9. Yes __ No__ N/A__ Are alternative representations of the menu topics available (e.g., expert-novice or alphabetical-topic organizations)?

10. Yes __ No__ N/A__ Can you select items by mouse or by entering a single letter or number?

11. Yes __ No__ N/A__ If the cursor must be moved to the selection, do you have full cursor control?

12. Yes __ No__ N/A__ Does it take less than 3 seconds to move between two related topics.

CATEGORY IV: FORMAT

Does the format of the help text facilitate searching for and understanding the needed information?

FORMAT TOTAL: Yes _____ No _____

1. Yes __ No__ N/A__ If there is more than one screen of help text do you move by paging?

2. Yes __ No__ N/A__ Do all or almost all screens have the same format?

3. Yes __ No__ N/A__ Is there a "strong" structure, i.e., a lot formatting?

4. Yes __ No__ N/A__ Are lists of three or more items presented in a list format (rather than as a string in a sentence or paragraph)?

5. Yes __ No__ N/A__ Does it appear that at least 50% of the screen is given to margins and space between units of information? (That is, do you perceive screen density to be 50% or less?)

6. Yes __ No__ N/A__ Are the chunks of content (syntax, function, example, etc.) clearly separated by spacing?

7. Yes __ No__ N/A__ Are headings or some form of highlighting used to identify the different chunks of information in the help?

8. Yes __ No__ N/A__ Is the typeface at least 10 point and is the font easy to read?

9. Yes __ No__ N/A__ Is there a mixture of upper and lower case letters?

CATEGORY V: CONTENT

What information is presented in the help text?

CONTENT TOTAL: Yes _____ No _____

1. Yes __ No__ N/A__ Is the help information focused on tasks and how to use commands to complete tasks?

2. Yes __ No__ N/A__ Is the information on options organized by tasks? For example, consequences of options are on the left and the options are on the right and the organization is based on similar option actions rather than the alphabetical order of option symbols?

3. Yes __ No__ N/A__ Does the help provide an actual example of the command syntax with options entered for you to model (not just a generic listing of the command syntax)?

4. Yes __ No__ N/A__ Does the help describe the function of the command?

5. Yes __ No__ N/A__ Are both quick reference and more detailed help information available (they could be available within a single help text through effective organization and formatting)?

6. Yes __ No__ N/A__ Are expert and novice help both available (more technical and explanatory vs more procedural)?

7. Yes __ No__ N/A__ Is a concrete example of using the command or command sequence provided in sufficient detail that you could model it?

8. Yes __ No__ N/A__ Are all possible applications or uses of the command or action listed?

9. Yes __ No__ N/A__ Is troubleshooting and warning information provided, e.g., information on common errors people make or how to get out of it if it doesn't work, or alternate paths to the goal if it doesn't work?

10. Yes __ No__ N/A__ Can you select a tutorial to provide additional assistance on a topic?

11. Yes __ No__ N/A__ Can you see a simulation of the task activity?

CATEGORY VI: COMPREHENSIBILITY

How clearly is the help text written?

COMPREHENSIBILITY TOTAL: Yes _____ No _____

1. Yes __ No__ N/A__ Is the help information easy to understand while browsing through it (i.e., reading it without a particular task in mind)?

2. Yes __ No__ N/A__ Are the sentences generally simple in syntactic structure (e.g., a simple sentence with no more than two clauses)?

3. Yes __ No__ N/A__ Are the sentences generally in active voice?

4. Yes __ No__ N/A__ Do lists of three or more items have a parallel structure?

5. Yes __ No__ N/A__ Is it very rare to find three or more nouns strung together?

6. Yes __ No__ N/A__ Is the presentation addressed to the reader through the use of personal pronouns?

7. Yes __ No__ N/A__ Is the vocabulary, beyond the command names, nontechnical and free of jargon?

8. Yes __ No__ N/A__ Does the help include graphics to show location or relation information?

9. Yes __ No__ N/A__ Can you easily identify the objects in the graphics, either through their appearance or through labelling?

10. Yes __ No__ N/A__ Are the objects in the graphics sized appropriately relative to each other?

CATEGORY VII: NAVIGATION

How easy is it to navigate between help texts?

NAVIGATION TOTAL: Yes _____ No _____

1. Yes __ No__ N/A__ Is there help on help?

2. Yes __ No__ N/A__ Is help on help readily available and clearly identified when you first enter the help system?

3. Yes __ No__ N/A__ Does the help on help provide an overview on the help system and strategies for using it in addition to the basic navigation commands?

4. Yes __ No__ N/A__ Is there an overview on the program for which the help was written?

5. Yes __ No__ N/A__ Does the overview of the program for which the help was written provide a scheme or structure for helping you think about the overall structure or function of the that program?

6. Yes __ No__ N/A__ Does the overview on the program for which the help was written give strategies and hints for effectively using that program?

7. Yes __ No__ N/A__ Can you move directly between related help topics (do not need to go to a menu)?

8. Yes __ No__ N/A__ Are related topics identified in the help text on a topic?

9. Yes __ No__ N/A__ Are subtopics identified in the help text on a topic?

10. Yes __ No__ N/A__ Can you select related topics or subtopics from the help text on a topic, i.e., is there a hypertext or embedded menu system?

11. Yes __ No__ N/A__ Is a keyword system available to facilitate navigation when you know the command or topic?

12. Yes __ No__ N/A__ Does it take less than 3 seconds to move between two related help topics?

CATEGORY VII: NAVIGATION (CONTD.)

13. Yes __ No__ N/A__ Is a bookmarking system available so that you can mark a place to return to after exploring?

14. Yes __ No__ N/A__ Are the prompts for how to navigate (i.e., the options of next command that can be issued) clearly visible and identifiable at all times?

15. Yes __ No__ N/A__ Is there an emergency procedure for recovering from getting lost, e.g., a command or selection to return to the main menu?

16. Yes __ No__ N/A__ Can you access a map of the help system showing your current location?

17. Yes __ No__ N/A__ Can you preview a topic or see some context on a topic before moving to it?

18. Yes __ No__ N/A__ Is the help system interactive, asking you for information and using that to determine what help to present?

CATEGORY VIII: LINKAGE

Does the link between the help system and the application software facilitate the application of the help.

LINKAGE TOTAL: Yes _____ No _____

1. Yes __ No__ N/A__ Can you transfer help information to the application program, for example, through cut and paste?

2. Yes __ No__ N/A__ Can you view the relevant portion of the application program while help is on the screen?

3. Yes __ No__ N/A__ Can you work on the application while help is on the screen?

4. Yes __ No__ N/A__ Does the functioning of the help system mimic the functioning of the application software (e.g., same movement commands, same kind of menu system, etc.)?

The Design

of

Online Help Systems

A Design Evaluation Instrument (B)

developed by

Thomas M. Duffy
Communications Design Center,
Carnegie Mellon University
Pittsburgh, PA 15213

1 March 1988

This work was funded in part by the United States Army
Human Engineering Laboratory, Aberdeen Proving Grounds, MD
under contract DAAA1-86-K-0019.
Contract monitor: Maureen Larkin.

Overview

This survey is for the evaluation of the quality of the design of any online help system. (An online help system is a computer program that provides job aiding information on the use of another program called the application software.) This component of the help evaluation process presumes that the help information is accurate and complete.

Eight design features of the help system are evaluated: support for problem representations, access to help, menu structure, the format of the help text, the kind of content presented in a help text, the comprehensibility of the content, navigation through the help system, and the linkage between the help system and the application software. The rating of each of these features is based on the amount of cognitive, perceptual, and motor effort that segment of the design imposes on the user, e.g., how much work and time does it take to access the help system.

The questions focus on the weaknesses of the help system. Thus the higher the raw score the poorer the help system. We convert the raw score for each component into a component score that is positively related to quality, i.e., a high "component score" indicates that that component of the help is well designed. The "System Score" is the overall rating of the help system and is the average of the component scores. Note that each component is weighted equally in calculating the System Score

Instructions

1. Complete the Evaluation Background on the next page.

2. Complete the questions in each of the eight Evaluation Categories.

   A.. Score the help system on every question in each of the eight Evaluation Categories. Circle the option under each question that best describes the help system you are evaluating.

   B. Add the circled numbers in a Category and enter that sum under "Score" at top of the first page for that Evaluation Category.

   C. Calculate the maximum score (maximum negative score) the software could receive in that Evaluation Category. The maximum for a question is generally the number of alternatives for that question. Thus the maximum for the Category will be the sum across questions of the number of alternatives. However, this may vary as a function of the complexity of the help system being evaluated. That is, if it is unreasonable for a help system to have a particular feature then it is unreasonable to give a large negative score to the system for not having the feature. *If an adjustment to the maximum score on a* question may be necessary the nature of the adjustment and the condition determining it, is presented in bold in the question stem. For example, in question A of Category III, the maximum is 1 if the help system has fifteen or fewer commands. Thus, in summing the total possible negative score for Category III, item A would add:
   one point if the help system had fifteen or fewer commands;
   three points if the help system had more than fifteen commands.

3. Transfer the obtained score and the worst possible scores recorded at the start of each Evaluation Category to the appropriate space in the Evaluation Summary on the next page.

4. Calculate the score for each component and the overall System Score. A well designed component well have a score near 1.0 and a well designed system will have a score approaching 100. Use the System Score and the Component Scores to compare help systems.

Evaluation Background

Name:_____ Version Number:_____

Manufacturer:_____

System it was tested on: _____

Evaluator: _____ Date:_____

Evaluation Summary

	A. Category Score	B. Worst Score Possible	C. Best Score Possible	E. A/B	F. 1.0 - 100.0 (Component Score)
Represent.	_____	6	2	____	____
Access	_____	14	5	____	____
Menus	_____	____ (22-15)	7	____	____
Navigation	_____	____ (18-10)	5	____	____
Content	_____	____ (15-9)	4	____	____
Comprehens.	_____	____ (27-23)	9	____	____
Format	_____	____ (20-14)	8	____	____
Linkage	_____	11	4	____	____

SYSTEM SCORE INSTRUCTIONS: **a).** Sum column E; **b).** divide by 7; **c).** multiply by 100.

The score may range between 100 (best system) and 0 (worst system).

SYSTEM SCORE: _____

CATEGORY I: PROBLEM REPRESENTATION

Does the help system support different naming for tasks and commands?

REPRESENTATION SCORE:_____ BEST SCORE: 2 WORST SCORE: 5

A. Are alternative menu systems available to reflect differing goals in entering help (e.g., alphabetical to support searching for a particular command; tasks to reflect real world tasks; and tasks to reflect computing tasks; or expert and novice organizations)?

1. Yes alternate representations are available and the alternate organizations make a lot of sense.
2. Yes alternate representations are available but one or more of the alternative organizations does not appear to be very helpful.
3. There is only one menu system but there are tasks and commands on it.
4. There is only a command menu.

B. Is there a keyword system or an index to support a range of synonyms for task and command names used in the application?

1. Yes.
2. No.

CATEGORY II: ACCESS

How easy is it to access the help system?

ACCESS SCORE:_____ BEST SCORE: 5 WORST SCORE: 14

A. Where must you be to access help?
1. Anywhere in the program.
2. Only in particular places, e.g, home base.
3. Must first exit the application or current activity.

B. How do you access help?
1. A continuously displayed and meaningful prompt.
2. A menu item but it is not continuously displayed on the screen; or, a continuously displayed prompt that is hard to interpret.
3. A command that is intuitively obvious or a "standard" but not continuously available.
4. A command that is not intuitively obvious or traditional and is not continuously available.

C. How long does it take to access the help system?
1. Two seconds or less.
2. Three to eight seconds.
3. More than eight seconds.

D. Does the help place you in a relevant subset of help?
1. Yes, placed in relevant help based on current context(current mode), user expertise, or user history.
2. No, placed in the same subset of help regardless of current context.

E. Does the help prompt you to seek help when you commit errors?
1. Yes.
2. No.

CATEGORY III: MENUS

How easy is it to locate information in the menus?

MENU SCORE:_____ BEST SCORE: 7 WORST SCORE: 22-15

A. How many items are on the top level menu? **(1 max if 15 or fewer items in entire help system)**
   1. 15 to 50; or, there are fewer than 15 items in the entire help system.
   2. 51 to 70; or, 8 to 14 if there are more than 15 items in the full help system.
   3. >70; or, <8 if there are more than 15 items in the full help system
   4. There are no menus; just a keyword system.

B. If there is a menu hierarchy (more than a top level menu), what is the average number of topics on the submenus? (including embedded menus)? **(1 max if no hierarchy)**
   1. between 7 and 15; or, no menu hierarchy is used.
   2. <7 or >15.

C. Are all items on a menu visible at once?
   1. Yes.
   2. No, the menu items are scrolled or paged through a window.

D. How are submenus and higher level menus linked? **(1 max if no submenus)**
   1. Submenus can be seen without leaving the higher level menu (e.g., through the use of pop up windows) or no submenus are used.
   2. Submenu is embedded in the help text for any item on the next higher menu (e.g., through hypertext or as a list at the end of the help information).
   3. Submenus replace higher level menus (the menu selections are a path to help text rather than help being provided on all menu entries.

E. How are the items on the menu organized?
   1. There is some organization of the menu items based on shared relevance (e.g., used in same task, part of the same topic, etc.) and there are headings for the groups.
   2. Menus are grouped as in 1, but there are no headings for the groupings.
   3. Items on the menu are presented alphabetically.
   4. Menu items have neither an alphabetical nor a common use organization, e.g., they may be organized by frequency of use or the organizational structure may not be apparent.

CATEGORY III. MENUS (CONTD.)

F. How do you select menu items?
   1. Click on item with a mouse or type a letter or number associated with the item.
   2. Move cursor to selection with full (up-down and left right) cursor control.
   3. Type first letter or letters of item to be selected.
   4. Make selection with the cursor in more than one column of entries without full cursor control.

CATEGORY IV: FORMAT

Does the format of the help text facilitate searching for and understanding the needed information?

FORMAT SCORE:_____ Best Score: 8 Worst Score: 20-14

A. If the help text takes up more than one screen, how do you move between screens? **(1 max if the help text never requires more than 1 screen)**
   1. Page; or text never requires more than one screen).
   2. Scroll.

B. How are lists formatted? **(1 max in the unlikely event that no lists are presented)**
   1. Usually presented as a bulleted or numbered list; or there are no lists of information.
   2. Usually presented as sentences or phrases in a paragraph.

C. How much of the screen is given to margins and spacing between chunks of information (do not include spacing between lines in your estimate)?
   1. Looks like at least 50%.
   2. Looks like between 30% and 50%.
   3. Looks like less than 30%.

D. Are the chunks of content (syntax, function, example, etc.) clearly separated by spacing? **(1 max if only one chunk in the help text)**
   1. Frequently; or, just one type of information presented.
   2. Sometimes.
   3. Infrequently (they are typically all mixed together).

E. Are headings or some form of highlighting used to identify the different chunks of information in the help? **(1 max if only one chunk in the help text)**
   1. Frequently; or, sparsity of help information makes highlighting unnecessary.
   2. Sometimes.
   3. Infrequently.

F. Are there basic organization and format principles applied consistently across the different help texts?
   1. A consistent organization and format are applied across most help texts -- the appearance (layout and organization) looks the same for the different help texts.
   2. A few rules are applied consistently.
   3. No rules are applied consistently.

CATEGORY IV: FORMAT (CONTD.)

G. Is the type face large enough (at least 10 point) and is the font easy to read?
   1. Yes.
   2. No.

H. Is there a mixture of upper and lower case letters?
   1. Yes.
   2. No.

CATEGORY V: CONTENT

What information is presented in the help text?

CONTENT SCORE:_____ Best Score: 4 Worst Score: 15-9

Note: Worst possible score is based on the assumption that only one entry in item D may be necessary.

A. Is the content task oriented?
   1. Focus is on tasks and how to use the commands in that task.
   2. Focus is on describing command syntax and function, but some attention is given to tasks or examples of applications.
   3. Help only describes command syntax, function, and related information (e.g., bugs), and an example of how to enter the syntax is given, but there is no (or trivial) task/application information. Or syntax is not relevant and the system only describes the function of the command.
   4. Help only describes the syntax or function without giving an example of the way in which the syntax would be entered.

B. Is the help system interactive, asking you for information and using that to determine what help to present?
   1. Yes.
   2. No.

C. Are there levels of explanation (e.g., quick reference vs elaborated or expert vs novice or simplified vs technical)?
   1. Yes.
   2. No.

D. What kind of help information is presented? For each category given below give one point if the category is always absent but COULD be there; 1/2 point if it is absent on some of the helps when it should be there.
   - syntax
   - function
   - list of related commands
   - possible applications
   - a concrete example of how the command is used presented in sufficient detail so that the user can imitate it.
   - bugs and warnings and troubleshooting
   - tutorials are available from the help

CATEGORY VI: COMPREHENSIBILITY

How clearly is the help text written?

COMPREHENSIBILITY SCORE:__ Best Score:9 Worst Score:27-23

A. How easy is it to understand (not apply) the help information while browsing through it?
   1. Very easy to understand.
   2. Somewhat difficult to understand.
   3. Very difficult to understand.

B. Are sentences overly complex in structure?
   1. Usually a simple structure.
   2. Usually a somewhat complex structure.
   3. Usually a very difficult sentence structure.

C. Are sentences in passive voice?
   1. Infrequently.
   2. Somewhat.
   3. Frequently.

D. Do lists have parallel structure?
   1. Always.
   2. Sometimes.
   3. Hardly ever.

E. Are noun strings used?
   1. Infrequently.
   2. Somewhat.
   3. Frequently.

F. Is the presentation addressed to the reader through use of personal pronouns?
   1. Frequently.
   2. Somewhat.
   3. Infrequently.

CATEGORY VI: COMPREHENSIBILITY (CONTD.)

G. Is the vocabulary, beyond the use of command and task names, unnecessarily difficult or technical?
   1. It is easy to understand.
   2. There is some unnecessary complexity.
   3. It is very difficult to understand the vocabulary.

H. Are graphics used to show location or relation information? **(1 max if no graphics are appropriate.)**
   1. Yes, consistently; or no graphics are appropriate.
   2. Yes, but only occasionally.
   3. No.

I. Are functional graphics (not graphics used just for motivation or to stimulate interest) easy to understand? **(1 max if no graphics are appropriate)**
   1. Very easy to understand; or, no graphics are appropriate.
   2. Reasonably easy to understand in general.
   3. Very difficult to understand; or, no graphics are used, but they should have been used.

CATEGORY VII: NAVIGATION

How easy is it to navigate between help texts?

NAVIGATION SCORE:_____ BEST SCORE: 5 WORST SCORE: 18-10

A. How helpful is the help on help? **(1 max if system is so simple that help on help is not needed)**

1. Very complete and clear -- gives overview and detailed hints and is readily available (i.e., there is a prompt for accessing help on help and that prompt is visible whenever the user enters help.); or help system is so simple that help on help is not required.
2. Help on help provided but does not describe the organization of the help or else it does not give strategies for use. Or the help on help is contained as a separate file that must be accessed outside of help.
3. No help on help provided.

B. How helpful is the overview on the application software?

1. The thorough overview provides a conceptual model to assist in thinking about the software, hints on strategies and potential pitfalls.
2. The overview provides some useful detailed information but does not provide an overview that helps in thinking about the software.
3. The overview information does not provide useful detail or an overview that helps in thinking about the software.
4. There is no overview on the application software.

C. How do you move around in the help? **(1 max if small system, e.g., 15 commands or less)**

1. There is a network of submenus and help that you can navigate through - going directly to related topics or going to a menu of related topics; or system is so small that only one menu is appropriate).
2. There is a single menu but you can go from anyone topic to another without returning to the menu through either a keyword or a "see also" listing in the help.
3. There is a single menu and the menu must be used to access any help topic to go to new help (i.e., no navigating permitted.)
4. There are multiple menus, but there are only certain fixed paths through the menus or help database (e.g., you must return through a path of two or more items/menus before taking another route).

CATEGORY VII: NAVIGATION (CONTD.)

D. How long does it take to move from one help topic to another (within the same general topic area if the help is broken into topic areas)?
   1. Less than 2 seconds.
   2. 2 to 5 seconds.
   3. more than 5 seconds.

E. What support is available for navigating? **(1 max if the system is so simple that it is impossible to get lost (e.g, fewer than 15 help entries or a single main menu that the user always goes through)**

   1. Four or more of the following navigation supports are available (or the system is so simple that navigation is not an issue):
      - bookmarking;
      - the requirements for navigating are always visible or the method for accessing the requirements is prompted;
      - emergency procedures for recovering from getting lost;
      - a map of the help system showing the user current location;
      - the ability to preview a topic or see the context surrounding the discussion of a topic.
   2. Two or three of the four supports are available.
   3. One of the supports is available.
   4. None of the supports is available.

CATEGORY VIII: LINKAGE

Does the link between the help system and the application software facilitate the application of the help.

LINKAGE SCORE:_____ Best Score: 4 Worst Score: 11

A. Can you transfer help information to the application program, e.g., through cut and paste?
   1. Yes.
   2. No.

B. Can you view the relevant portion of the application program while help is on the screen?
   1. Yes.
   2. Sometimes (Overlapping window with fixed placement).
   3. No.

C. Can you work on the application while help is on the screen?
   1. Yes, and this is true for all help information.
   2. Yes, but only for a reduced portion of the help information.
   3. No.

D. Does the functioning of the help system mimic the functioning of the application software (e.g., same movement commands, same kind of menu system, etc.) ?
   1. Yes, fully or almost fully mimics the application software.
   2. Somewhat.
   3. No.

Finding Information on a Menu: Linking Menu Organization to the User's Goals

(Pilot Study Data and Experiment in Progress)

developed by

Brad Mehlenbacher, Thomas M. Duffy, James Palmer.

Communications Design Center
Carnegie Mellon University
Pittsburgh, Pa. 15213

1 March 1988

This work was funded in part by the United States Army
Human Engineering Laboratory, Aberdeen Proving Grounds, MD
under contract DAAA1-86-K-0019.
Contract monitor: Maureen Larkin.

Abstract. Although research on how best to organize menu items to facilitate learning, search speed and reduced selection errors is pervasive, very little has been done to examine the effect of different types of user tasks on a menu's effectiveness. In a pilot study using three, distinct task types and two menu organizations, alphabetic and functional ordering, preliminary data suggests that (1) the more difficult a task is the more effective a functionally organized menu becomes, (2) learning occurs with both menu designs, more notably with the alphabetic ordering, although the functional ordering overall takes less time to make selections from, (3) the time spent looking at the tasks decreases across repeated trials, and (4) while users made fewer errors with the functional design overall, they corrected more across trials using with the alphabetic design. The results, in general, encourage more rigorous investigation of the interaction between the tasks users bring to menu interfaces, and the optimal design of those menus.

1. Introduction

Among the various methods designed to facilitate human-computer interaction, menu-driven interfaces have received an immense amount of use and attention over the last ten years (Giroux & Belleau, 1986). The addition of menus to existing software, with the guarantee of increased system friendliness, is a standard development in the computer industry; so the essential difference between, for example, Unix and Andrew, CMS and PROFS, Dbase II and Dbase III, is the addition of a menu(s) -- their functionality is relatively unchanged, only the method of accessing the system's functions has been "enhanced".

In designing the menu interface, choosing the appropriate organization of the menu items is a crucial consideration. The arrangement of menu items effects the amount of time users spend searching for items (Allen, 1983), the number of correct selections made (Lee et al., 1986), how easily the menu can be learned (Bauer & Eddy, 1986), and the overall user satisfaction with the product (Magers, 1983).

Having established that a system's menu can have a significant impact on that system across various measures of effectiveness, researchers have begun manipulating several organizational features of menus. The goal of any menu should be to facilitate, in the best possible fashion, the user's ability to make a choice quickly and accurately.

The traditional method of organizing menu items is alphabetic, which is imposed on the user, machine-oriented and probably based on the pragmatics of implementation; the assumption is that "everyone" comprehends alphabetic structuring. If the user knows that the command he wants to select starts with the letter "S" (for example, SAVE), he looks three-quarters of the way down the alphabetic menu list. The problem with alphabetic organization, however, is that if the user does not know what he is looking for, alphabetic ordering offers no "clues" about the location of the menu item. That is, if we know we want to ZAP text, and the correct command word is DELETE, when we do not find the correct word near the end of the list, where Z-words should be located, we are without a method of approaching the menu organization without merely scanning left-to-right, top-to-bottom. Since the rationale for menu-driven interfaces is that they allow users to access a program's functions without having to recall, precisely, the exact command syntax, an alphabetic ordering would seem most effective only when we know the exact word for which we are searching.

According to Card's findings (1982) "initial search is faster for an alphabetic menu than for a functionally arranged menu or especially for a random menu." Also, after a user becomes familiar with the menu, the order of the menu items has no effect on search speed and all ordering methods are equally efficient. It is not surprising that random ordering scored the poorest in terms of selection time, since users would be forced to scan from top to bottom for each menu item; however, one would assume that after several repeated trials, even the random design would become familiar to the user, and selection time would decrease radically compared to the overall decrease for either the alphabetic or functional designs. So Card's conclusion that all menu organizations are equally learned seems counter-intuitive. Despite this, his research on menu-based command systems and user search mechanisms is frequently cited by menu presentation researchers and has encouraged industry's use of alphabetic ordering for command menus.

Yet several findings conflict with Card's stance. Shneiderman (1986), for example, cites the research findings of McDonald et al. (1983) and Somberg & Picardi (1983) as suggesting that semantic or functional ordering is superior to alphabetic for searching and selecting menu items. Using menus with sixty-

four items, McDonald et al. questioned the "ecological validity" of past menu organization research, stating that since "menu-based interfaces are most appropriate when recall is imperfect and the user must rely on item recognition", explicit targets, where a word is given to the user and the user is expected to select that same word from a menu, are unrealistic. Rather, tasks are often complex and fuzzy in the transfer from the schema the user brings to the software interface, and the actual command syntax at hand; in short, users do not always know what command will allow them to achieve their goal. McDonald et al. propose that it is more common for users to know what they want to do, "but not [to] know the exact function name ... to accomplish the task. They conclude their research arguing that "researchers in this area should focus more on task behavior more dependent of schema formation, such as problem solving, rather than simply menu access time." Giroux and Belleau (1986) also conclude that "searching a menu may or may not proceed sequentially and [that] the optimal organization may be semantic versus alphabetical".

Functional- or Task-oriented organizational design is a recent phenomena (Bethke et al., 1981). The premise is that systems, and thus their menu displays, should be user-oriented, goal-based, modelled (ideally) on human performance/understanding levels (Kieras, 1985); that the user is more important than the machine's functionality as a design issue. A functionally ordered menu for a word processing program, for example, would separate menu items into functional groups such as formatting, moving text, working with files, and so on. The alphabetic method of ordering menu items tends to assume that searching is accomplished by simply scanning from left to right, top to bottom, using alphabetic "cues" to guide the user's progress towards the accurate item. Functionally organized menus should provide "perceptual chunks" (Chase & Simon, 1973) of information that are more directly related to the actual task the user is trying to accomplish.

In general then conclusions about how best to organize menu items are often conflicting. The emphasis on the menu ordering rather than on the various and complex real world tasks users want to achieve may be the actual problem. Although research on menu order is pervasive, very little has been done to examine the effect of the task on a menu's effectiveness. Human

Factors researchers have, however, stressed the need for appropriate user and task models (Rissland, 1984; Rubenstein & Hersh, 1984; Kieras, 1985; Fowler & Roeger, 1986), so it is surprising that different task types, especially those which are difficult to transfer from user goal to interface command, have not been manipulated in more menu experiments. Would a more difficult or unclear task have an effect menu search time or selection accuracy? Does a menu's effectiveness depend on the goal and task of the user?

Researchers have tended to concentrate on perceptual matching tasks (Card, 1982; Lee & MacGregor, 1985), ignoring other, perhaps more applicable task types. And Landauer and Golotti (1984) point out that "researchers who aim to uncover the cognitive components of name learning tend to use contrived, not real, tasks". For this reason, we want to use a generic word processing menu, which contains many of the typical word processing commands found on menus.

1.1 Interaction between task type and menu organization

Our major interest is in creating three tasks that represent common word processing activities:

- searching for a target word. That is, "I know the word that the application uses and I want to find it";

- searching for a synonym match. That is, "I know what I want to do, but do not know the exact word the application uses", and;

- searching with a visual idea (before-after pictures). That is, "I have a picture of what I want the document to look like and I want to find the command that will allow me to accomplish it".

I am assuming that the time spent looking at each task type will increase with the difficulty of the task. My hypotheses are that more difficult (or fuzzy) tasks will be easier to accomplish using the functional menu, and that, with repeated trials, learning will occur and selection time will decrease generally. Direct word matching, which Card (1982) explored, will be faster and more

accurate with the alphabetic menu. If the goal of functional ordering is achieved, users should have more clues for selecting the correct item with less well-defined tasks, and make fewer selection errors. By manipulating the task type, we hope to show that one reason for inconsistent findings in menu organization research is that the goal or task of the user factors into the selection process.

2. Design

Two experimental variables are arranged in a 2 x 3 factorial design. The first variable, menu organization, consists of

- items organized alphabetically on the menu, and;
- items organized functionally on the menu.

The second variable, task difficulty, consists of

- a direct match task type;
- a synonym task type, and;
- a before-after picture task type.

Each cell, for the purposes of this pilot, contained two subjects who were randomly selected for the various conditions. Figure 1 displays the design graphically.

Group	Condition 1	Condition 2
1	Alphabetic Menu	Direct Match Task
2	" "	Synonym Task
3	" "	Picture Task
4	Functional Menu	Direct Match Task
5	" "	Synonym Task
6	" "	Picture Task

Figure 1. Experimental groups by condition combinations.

2.1 Materials

The entire experiment is being conducted using a Macintosh Plus terminal and mouse, and was designed with HyperCard, a software application program. Subjects are presented with the instructions, a task, and with the selection menu online. Both menus, the alphabetic and functional (see Appendix 4), have twenty-two items in three columns on them. The functional menu also has six headings (or functionally distinct word processing categories). The number of items we have used will balance, we expect, the range set by Card who had eighteen items in one column and McDonald et al. who had sixty-four items in four columns. Confronted with so much to choose from on the McDonald et al. menu, the user would probably rely more on functional groupings to cue him in his scanning for the correct choice. With Card's menu however, given direct matching tasks and low density of information on the actual menu, single, top-down scanning would prove successful. My menu size reflects many of the conventional word processing menus with between fifteen and twenty-five items to choose from (see, for example, WordStar, MicroSoft Word, EZ, etc.).

3. Results of Pilot Study

A 2 x 3 x 5 ANOVA (Menu type x Task type x Blocks of trials) was performed on each dependent variable. Since it was a pilot study, with a very limited N (two subjects per cell), Alpha was accepted at .05 or less. For each dependent variable, the following statistically significant, and near significant, effects were obtained:

3.1 Time spent looking at the task

Figure 2 shows a graph of the mean time subjects spent looking at each of the three types of tasks across the five trial blocks. Not surprisingly, the number of trials was significant, $F(4,24) = 14.827$, $p < .000$, and the interaction between the number of trials and the task type was significant, $F(8,24) = 5.462$, $p < .001$. As well, factoring with a logarithmic function results in a correlation of .97.

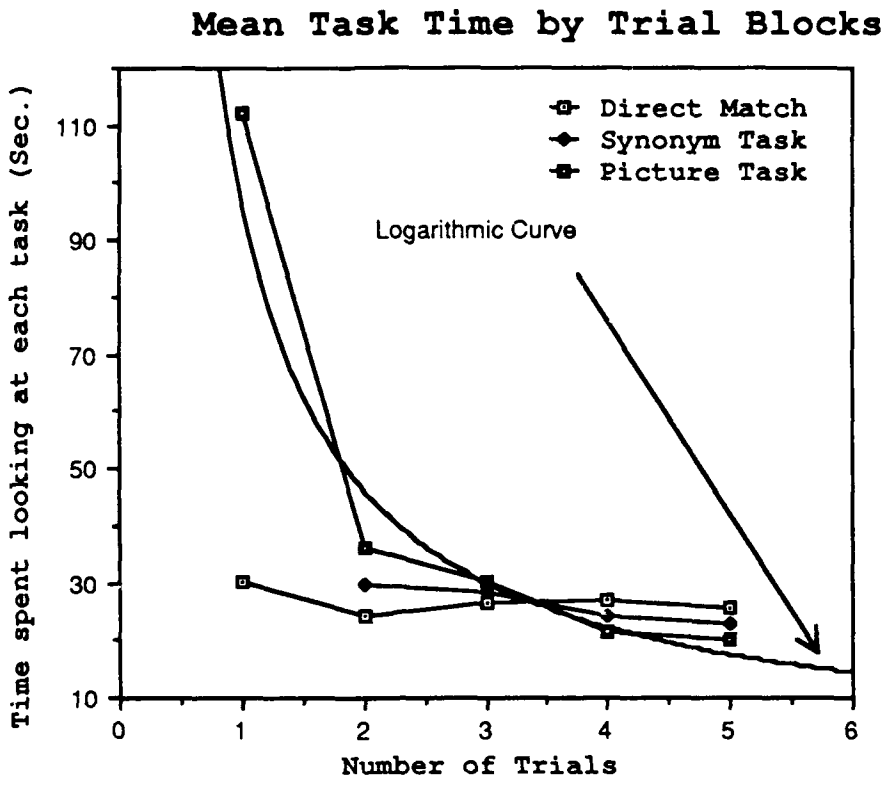


Figure 2. Mean time spent looking at task types by trial blocks.

3.2 Menu selection time

Again, the number of trials was significant, $F(4,24) = 26.639$, $p < .000$. The interaction of menu type by task type approached significance, $F(2,6) = 3.632$, $p < .093$, as did the interaction of trials by menu type, $F(4,24) = 2.477$, $p < .071$ (see Figure 4). The interaction between menu type, task type and number of trials also approached significance, $F(8,24) = 2.207$, $p < .064$. Factoring the menu type and selection time across trials using a logarithmic function produces a correlation of .96; alphabetic ordering x selection time x trials has a correlation of .98 and functional ordering x selection time x trials has a correlation of .96.

3.3 Number of incorrect selections

Figure 5 shows a graph of the number of incorrect selections made by subjects using the different menu types across trial blocks. Again, the number of trials

was significant, $F(4,24) = 4.286$, $p < .009$, and the interaction of the number of trials by the menu type was significant, $F(4,24) = 4.286$, $p < .009$.

4. Discussion

Considering the pilot study was run with an extremely small group of subjects, only two per experimental condition, the preliminary data is incredibly encouraging. Learning occurred across all three measures -- predictably, the time spent looking at each task, whether direct, synonym or picture, decreased in significantly over the five, repeated trials; similarly, learning occurred with the menu selection time and the number of selection errors. Subjects took the greatest amount of time on the first trial, whether in attempting to understand the task required of them or in searching for the appropriate menu item to achieve that task. And interestingly, when presented with the same task the second or third time, several subjects corrected their original selection choice.

Especially interesting was the near significance that the type of menu and the type of task had on the amount of time spent selecting a menu item. Graphically, it appears that when subjects were searching for a direct match task, the alphabetic organization took less selection time than the functional menu design (see Figure 3).

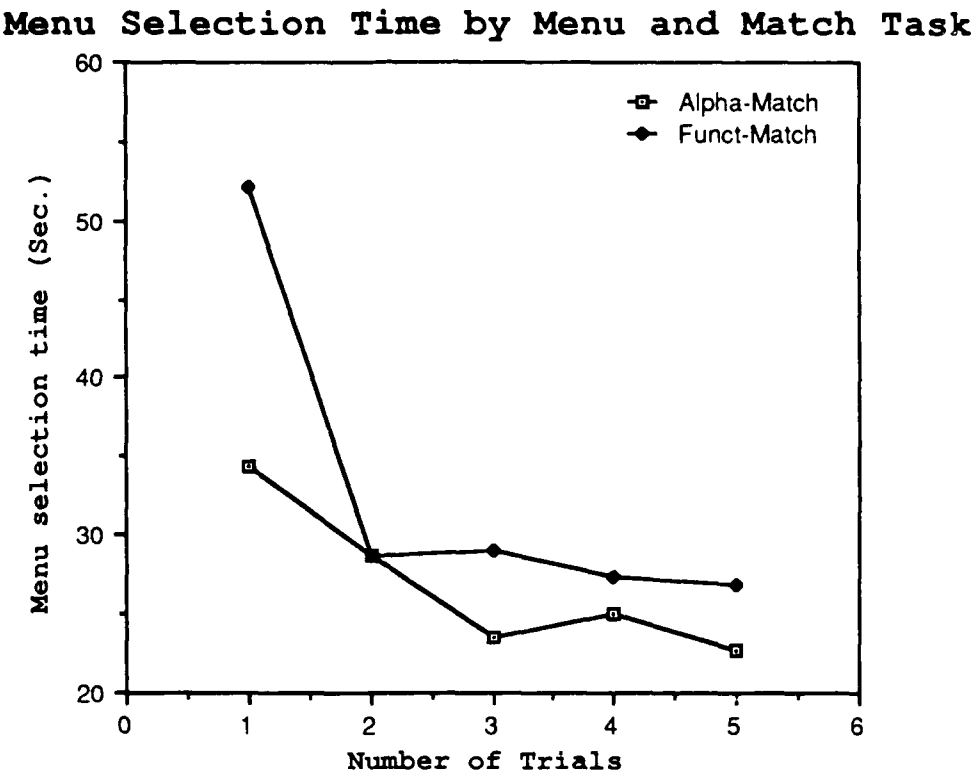


Figure 3. Mean menu selection time with the direct match task.

As Figure 4 shows, for the synonym matching task, the alphabetic and the functional organizations were relatively similar in selection time (decreasing similtaneously across the five trials). Although it is interesting that the selection time for the first trial is definitely better for the functional than the alphabetic organization.

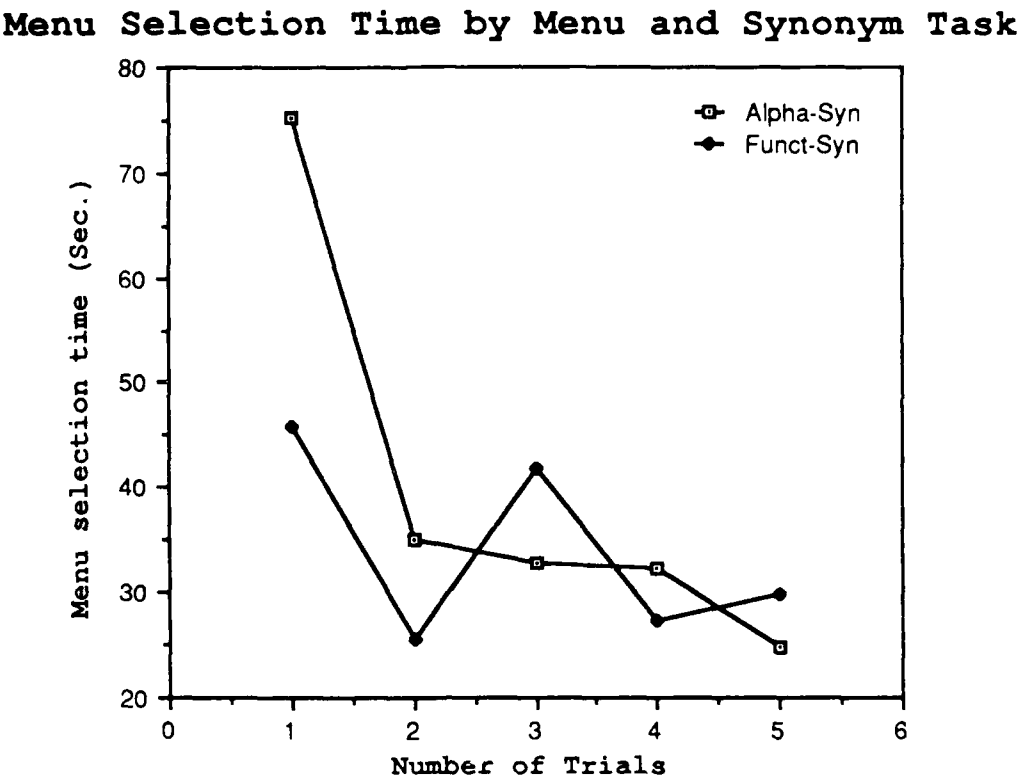


Figure 4. Mean menu selection time with the synonym task.

The most distinct difference between the selection times of the two menus occurs with the before-after picture task. This task is definitely the most difficult of three, which the number of selection errors across the three task types supports. With the final task, the functional menu had a lower selection time than the alphabetic menu across all five trials (see Figure 5).

Online Help References

by

Communications Design Center
Carnegie Mellon University
Pittsburgh, PA 15213

1 March 1988

This work was funded in part by the United States Army,
Human Engineering Laboratory, Aberdeen Proving Grounds, MD
under Contract # DAAA1-86-K-0019 AC85.
Contract monitor: Maureen Larkin

Al-Awar, J., Chapanis, A. & Ford, W. (1981, March). Tutorials for the first-time computer user. *IEEE Transactions on Professional Communication,* PC-24 (1), 30-37.

Allen, R. B. (1982). Cognitive factors in human interaction with computers. *Behaviour and Information Technology,* 1(3), 257-278.

----------. (1983). Cognitive factors in the use of menus and trees: an experiment. *IEEE Journal on Selected Areas in Communications,* SAC-1(2), 333-336.

Apple Computer, Inc. (1987). *Human Interface Guidelines: The Apple Desktop Interface.* Reading, MA: Addison-Wesley.

Armbruster, B. (1986). Schema theory and the design of content-area textbooks. *Educational Psychologist,* 21, 253-267.

Badre, A. N. (1981). Designing the human-computer interface. Association for Computing Machinery, 288-291.

Ball, E. & Hayes, P. (1981). *Representation of task-specific knowledge in a gracefully interacting user interface.* Carnegie Mellon, Computer Science Dept.

Bannon, L., O'Malley, C, Root, R, Norman, D., et al. (1983). User centered system design: papers for the CHI '83 conference on human factors in computer systems. *CHI '83* (ONR Report 8303).

Basara, D., Burgin, D., Ryan, G., & Trummel, P. (1986). A case study of online information: Second generation systems design. *IEEE Transactions on Professional Communications,* PC 29 (4), 81-86.

Bassok, M. & Holyoak, K. *Schema-based interdomain transfer between isomorphic algebra and physic problems*. Preliminary draft.

Bauer, D. & Eddy, J. (1986). The representation of command language syntax. *Human Factors*, 28 (1), 1-10.

Bethke, F. J. (1983). Measuring the usability of software manuals. *Technical Communication* (2), 13-16.

----------, Dean, P., Kaiser, E., Ort, E. & Pessin, F. (1981). Improving the usability of programming publications. *IBM Systems Journal*, 20 (3), 306-320.

Blaiwes, A. S. (1974). Formats for presenting procedural instructions. *Journal of Applied Psychology*, 59 (6), 683-686.

Boehm-Davis, D. & Fregly, A. (1985). Documentation of concurrent programs. *Human Factors*, 27 (4), 423-432.

Booher, H. R. (1975). Relative comprehensibility of pictorial information and printed words in proceduralized instructions. *Human Factors*, 17 (3), 266-277.

Bork, A. (1983). A preliminary taxonomy of ways of displaying text on screens. *Information design journal* 3/3, 206-14.

----------, Franklin, S., Von Blum, D., Katz, M., & Kurtz, B. *Graphics and screen design for interactive learning*. University of California, Educational Technology Center, Irvine.

Borenstein, Nathaniel S. (1985). *The Design and Evaluation of Online Help Systems*. PhD Dissertation, Pittsburgh, PA: Carnegie Mellon.

Brachman, R., Fikes, R., & Levesque, H. (1983). KRYPTON: A functional approach to knowledge representation. *IEEE Computer*, 67-73.

Bradford, Annette N. (1983, October). *Enhanced User Interface Through Computer Tutorials."* Paper delivered at the IEEE Conference on Professional Communications, Atlanta, GA, 131-134

Bradford, Annette N. (1984). Conceptual Differences Between the Display Screen and the Printed Page. *Technical Communication,* Third Quarter, 13-16.

Bransford, J., Sherwood, R., Vye, N. & Reiser, J. (1986). Teaching thinking and problem solving. *American Psychologist,* 41, 1078-1089.

Breuker, J. & de Greef, P. (1985, December). *Information Processing Systems and Teaching & Coaching in HELP Systems.* Deliverable 12.1, ESPRIT Project P280 (EUROHELP), Amsterdam: Department of Social Science Informatics, University of Amsterdam.

Britton, B.K. (1986). Capturing art to improve text quality. *Educational Psychologist,* 21, 333-356.

Brockman, J. R. (1986). *Writing Better Computer User Documentation.* New York: John Wiley & Sons.

Brooks, R. (1981). A theoretical analysis of the role of documentation in the comprehension of computer programs. Association for Computing Machinery, 125-129.

Brophy, J. (1986). Teacher influences on student achievement. *American Psychologist,* 21, 1069-1077.

Brown, P. J. (1985, September). Making Unix online documentation more effective. *Microprocessors and Microsystems,* 9 (7), 346-349.

Bunderson, C., Gibbons, A. & Olsen, J. (1981). Work models: beyond instructional objectives. *Instructional Science,* 10, 205-215.

Bury, K. Davies, S. & Darnell, M. (1985, June). *Window management: A review of issues and some results from user testing.* Unpublished

manuscript, IBM Corporation, General Products Division, Santa Teresa Laboratory, Human Factors Center.

Canter, D., Rivers, R. & Storrs, G. (1985). Characterizing user navigation through complex data structures. *Behaviour and Information Technology*, 4 (2), 93-102.

Card, S. K. (1982). User perceptual mechanisms in the search of computer command menus. *Proc. Human Factors in Computer Systems*, 190-196.

----------, Moran, T. P., and Newell, A. (1983). *The psychology of human-computer interaction*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Carroll, J. & Kay S. (1985, April). Prompting, feedback and error correction in the design of a scenario machine. *CHI'85 Proceedings*, 149-153.

----------. (1984, March). *Designing MINIMALIST training material.* IBM Watson Research Center, Research Report RC 10438 (#46643).

----------, & Thomas, J. *Metaphor and the cognitive representation of computing systems*, Yorktown Heights, N.Y.: IBM Research Report RC-8302.

----------, Mack, R. L., Lewis, C. H., Grischkowsky, N. L., & Robertson, S. R. (1985). Exploring exploring a word processor. *Human-Computer Interaction*, 1, 283-307.

Casey, B. E. (1986). A games approach to system interface design. *IEEE Transactions on Professional Communications*, PC 29 (4), 65-71.

Chase, W.G., & Simon, H.A. (1973). The Mind's Eye in Chess. *Visual Information Processing*, New York: Academic Press. 215-282.

Chavalaz, G., Tijerina, L., Modisette, L., Herschler, D. (1985, January). *Final report on menu and display literature search to U.S. Army Human Engineering Laboratory.* Battelle, Columbus Laboratories.

Claussen, C., Cohen, B., Halliday, P., Inman, E., Pfeil, J. & Phillips, M. *Computer-human operational requirements analysis system (CHORAS).* Computer Technology Associates, Englewood, Colorado.

Cohen, S. (1976). Speakeasy : A window into a computer. *National Computer Conference,* 1976, 1039-1047.

Cohill, A. & Williges, R. (1985). Retrieval of HELP information for novice users of interactive computer systems. *Human Factors,* 27 (3), 335-343.

Cox, G. & Walsh, B. (1980). A HELP system for the user community. *Software--Practice and Experience,* 10, 219-229.

Davis, E. & Swezey, R. (1983). Human factor guidelines in computer graphics: a case study. *Int. J. Man-Machine Studies,* 18, 113-133.

Davis, R. (1982, Spring). Expert systems: where are we? and where do we go from here? *The AI Magazine,* 3-22.

Dee-Lucas, D. & Larkin, J. (1986). Novice strategies for processing scientific texts. *Discourse Processes,* 9, 329-354.

Delisle, N. & Schwartz, M. (1985, December). *Hypertext users manual.* Technical Report #CR-85-55, Computer Research Lab, Tektronix Labs.

Demers, R. A. (1981, August). System design for usability. *Communications of the ACM* 24 (8), 494-501.

diSessa, A. A. (1986). Models of computation. D. A. Norman & S. W. Draper (Eds.), *User centered system design* (pp. 201-218). Hillsdale, NJ: Lawrence Erlbaum Associates.

Dray, S. M., Ogden, W. G., & Vestewig, R. E. (1981). Measuring performance with a menu-selection human-computer interface. *Proc. Human Factors Society, 25th Annual Meeting,* 746-748.

Duchnicky, R. & Kolers, P. (1983). Readability of text scrolled on VDTs as a function of window size. *Human Factors*, 25, 683-692.

Duffy, T., Trumble, J., Isenberg, T., Janik, C. and Rodgers, K. (1987). *Learning with Online and Hardcopy Tutorials.* Technical Report CDC-32 (ERIC # 284 562), Pittsburgh, PA, Communications Design Center, Carnegie Mellon.

Duffy, T. & Langston, D. (1985, November). *Online help: Design issues for authoring systems.* Technical Report CDC-18 (ERIC # ED-267-810). Pittsburgh: Communications Design Center, Carnegie Mellon.

Duffy, T. & Langston, D. (1985, March). *Online Help for Automated Training Development*. Pittsburgh: Carnegie Mellon.

Dunsmore, H. E. (1980). Designing an interactive facility for non-programmers. ACM, 475-483.

Dzida, W., Herda, S. & Itzfeldt, W. (1978). User-perceived quality of interactive systems. *IEEE Transactions on Software Engineering*, SE-4 (4), 270-276.

Ehrich, R., Hix, D., & Hartson, H. *Interactive tools: Making UIMS usable.* Submitted to ACM/SIGGRAPH Workshop on Software Tools for User Interface Development.

Elkerton, J. & Williges, R. (1984). Information retrieval strategies in a file-search environment. *Human Factors*, 26 (2), 171-184.

Elkerton, J. & Williges, R. (1985). A performance profile methodology for implementing assistance and instruction in computer-based tasks. *Int. J. Man-Machine Studies*, 23, 135-151.

Elm, W. & Woods, D. (1985, October). Getting lost: a case study in interface design. *Proceedings of Human Factors Society.* Westinghouse Scientific Paper: 85-1C60-CONRM-P2.

Engelbart, D. & English W. (1968). A research center for augmenting human intellect. *Fall Joint Computer Conference*, 1968, 395-410.

Engle, S. & Granda, R. (1975, December). *Guidelines for man/display interfaces.* Technical Report 00.2720, IBM Poughkeepsie Laboratory.

Fenchel, R. & Estrin, G. (1982). Self-describing systems using integral help. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-12 (2), 162-167.

Feurzeig, W. (1985, February). *AI aids for design and automation of CAI programs.* Carnegie Mellon University.

Folley, L. & Williges, R. (1981). User models of text editing command languages. Association for Computing Machinery, 326-331.

Fowler, S.L. & Roeger, D. (1986, December). Programmer and writer collaboration: Making user manuals that work. *IEEE Transactions on Professional Communications*, PC-29 (4), 21-25.

Frase, L. & Schwartz, B. (1979). Typographical cues that facilitate comprehension. *Journal of Educationai Psychology*, 7, 197-206.

Furnas, G. W. (1986, April). Generalized fisheye views. *CHI'86 Proceedings*, 16-23.

Furnas, G., Landauer, T., Gomez, L. & Dumais, S. (1986, July-August). Statistical semantics: analysis of the potential performance of key-word information systems. *The Bell System Technical Journal*, 62 (6), 1753-1806.

Gaylin, K. B. (1986, April). How are windows used? Some notes on creating an empirically-based windowing benchmark task. *CHI'86 Proceedings*, 96-100.

Gick, M. & Holyoak, K. (1983). Schema induction and analogical transfer. *Cognitive Psychology*, 15 (1-38).

Gick, M. & Holyoak, K. (1980). Analogical problem solving. *Cognitive Psychology*, 12, 306-365.

Girill, T. & Luk, C. (1983, May). Document: An interactive, online solution to four documentation problems. *Communications of the ACM*, 26 (5), 328-337.

Giroux, L. & Belleau, R. (1986). What's on the menu? The influence of menu content on the selection process. *Behaviour and Information Technology*, 5 (2), 169-172.

Gomez, L. M., Egan, D. E., & Bowers, C. (1986). Learning to use a text editor: some learner characteristics that predict success. *Human-Computer Interaction*, 2, 1-23.

Gould, J. & Grischkowsky, N. (1984). Doing the same work with hard copy and with cathode-ray tube (CRT) computer terminals. *Human Factors*, 26 (3), 323-337.

Gould, J. D. (1985, April). How to do psychological research on person-computer issues. *CHI'85 Proceedings*.

Gould, L. & Finzer, W. (1981). A study of TRIP: A computer system for animating time-rate-distance problems. In R. Lewis & D. Tagg (Eds.), *Computers in Education*. North-Holland Publishing Co.

Graves, M., Slater, W., Roen, D., Redd-Boyd, T., Duin, A., Furniss, D., & Hazetine, P. (1986). *Some characteristics of memorable expository writing: Effects of revision by writers with different backgrounds*. Paper presented at the National Reading Conference, Austin, TX

Gray, J. (1986, January). The role of menu titles as a navigational aid in hierarchical menus. *SIGCHI Bulletin*, 17 (3), 33-40.

Grudin, J. & Barnard, P. (1984). The cognitive demands of learning and representing command names for text editing. *Human Factors*, 26 (4), 407-422.

Haas, C. (1987). *Computers and the Writing Process: A Comparative Protocol Study*. Technical Report CDC-34 (ERIC # 281-219), Pittsburgh, PA, Communications Design Center, Carnegie Mellon.

Hammer, J. & Rouse, W. (1979, January). Analysis and modeling of freeform text editing behavior. *IEEE*, 659-664.

Hanson, S., Kraut, R. & Farber, J. (1984, March). Interface design and multivariate analysis of UNIX command use. *ACM Transactions on Office Information Systems*, 2 (1), 42-57.

Hartley, J. (1982). Information mapping: A critique. *Information design journal*, 3/1, 51-58.

----------. (1981, March). Eighty Ways of Improving Instructional Text. *IEEE Transactions on Professional Communication*, PC24(1), 17-27.

Hartson, H. & Hix, D. (1985, November). Human-computer interface development: concepts and systems for its management. *ACM Computing Surveys*.

Hartson, H., Johnson, D. & Ehrich, R. (1984, September). A human-computer dialogue management system. *Interact '84 (First IFIP Conference on Human-Computer Interaction)*. London, England, 57-61.

Hayes, P. J. (1982). Uniform help facilities for a cooperative user interface. *National Computer Conference*, 1982, 469-474.

Heines, Jesse M. (1984). *Screen Design Strategies for Computer-Assisted Instruction*. Bedford, MA: Digital Press.

Hiltz, R. & Turoff, M. (1982, January). Human diversity and the choice of interface: A design challenge. *SIGSOC Bulletin 13 Proceedings of the Conference on Easier & More Productive Use of Computer Systems, Part II: Human Interaction & the User Interface.*

Hodgson, G. & Ruth, S. (1985, July). The use of menus in the design of on-line systems: A retrospective view. *SIGCHI Bulletin*, 17 (1), 16-22.

Holland, M. & Rose, A. (1981, November). *A comparison of prose and algorithms for presenting complex instructions*. DDP Technical Report No. 17.

Holt, R., Boehm-Davis, D. & Schultz, A. (1986, April). The effects of structured, multi-level documentation. *CHI'86 Proceedings*, 122-127.

Houghton, Jr. R. C. (1984) Online help systems: A conspectus. *Communications of the ACM* 27(2), 126-132.

Hurd, J. C. (1982, May). Writing online help. *Proceedings of the 29th International Technical Communications Conference (ITCC)*, W&E-151 - W&E-154.

Hutchins, E. L., Hollan, J. D., & Norman, D. A. (1986). Direct manipulation interfaces. D. A. Norman & S. W. Draper (Eds.), *User centered system design* (pp. 87-124). Hillsdale, NJ: Lawrence Erlbaum Associates.

IBM (1985). *Automated Author Aiding Systems Conference (Final Report)*. Designing Task-Oriented Libraries for Programming Products.

Isa, B., Neal, A., Evey, R. & McVey, B. (1982). Roadmaps versus roadsigns: Which is the better metalanguge? *Proceedings of the Human Factors Society (26th Annual Meeting)*, 994-998.

Jackson, P. & Lefrere, P. (1984). On the application of rule-based techniques to the design of advice-giving systems. *Int. J. Man-Machine Studies*, 20, 63-86.

James, Geoffrey. (1985). *Document Databases: The New Publications Methodology*. New York, NY: Van Nostrand Reinhold.

Johnson, D. & Hartson, H. (1986, April). An interactive environment for dialogue development: Its design, use, and evaluation -or- Is AIDE useful? *CHI'86 Conference on Human-Computer Interaction* (Boston).

Judisch, J., Rupp, B. & Dassinger, R. (1981). Effects of manual style on performance in education and machine maintenance. *IBM Systems Journal,* 20 (2), 172-183.

Kalish, H.I. (1969). Stimulus generalization. In M. Marx (Ed.), *Learning: Processes*. London: McMillian Co.

Kennedy, T. C. S. (1974). The design of interactive procedures for man-machine communication. *Int. J. Man-Machine Studies,* 6, 309-334.

Kennedy, T. C. S. (1975). Some behavioural factors affecting the training of naive users of an interactive computer system. *Int. J. Man-Machine Studies,* 7, 817-834.

Kerr, S. T. (1986). Learning to use electronic text: An agenda for research on typography, graphics, and interpanel navigation. *Information design journal,* 4/3, 206-11.

Kieras, D. (1985). *Improving the Comprehensibility of a Simulated Technical Manual*. Technical Report No. 20, TR-85/ONR-20. Ann Arbor, Mich: University of Michigan.

Kieras, D.E. & Dechert, C. (1985). *Rules for Comprehensible Technical Prose: A Survey of the Psycholinguistic Literature*. Technical Report No. 21, TR-85/ONR-21. Ann Arbor, Mich: University of Michigan.

Kieras, D. E. & Polson, P. G. (1982, May). *An outline of a theory of the user complexity of devices and systems*. Working Paper No. 1 (U. of Arizona, U. of Colorado, IBM collaborative project).

Kieras, D. E. & Polson, P. G. (1983). A generalized transition network representation for interactive systems.. A. Janda (Ed.), *Proceedings of the CHI.*

Kiger, J. I. (1984). The depth/breadth trade-off in the design of menu-driven user interfaces. *Int. J. Man-Machine Studies*, 20, 201-213.

Kruk, R. & Muter, P. (1984). Reading of continuous text on video screens. *Human Factors*, 26 (3), 339-345.

LaJoie, S. P. (1986, April). *Cognitive task analysis: Implications for intelligent tutor development*. Paper presented at the AERA Annual Meeting (San Francisco, California).

Landauer, T.K., & Galotti, K. (1984). What makes a difference when? Comments on Grodin and Barnard. *Human Factors*, 26(4), 423-429.

----------, & Nachbar, D. W. (1985). Selection from alphabetic and numeric menu trees using a touch screen: breadth, depth, and width. *Proc. Human Factors in Computing Systems, ACM SIGCHI*, 73-78.

Ledgard, H., Andrew, S., Whiteside, J. (1981). *Directions in human factors for interactive systems*. In G. Goos & J. Hartmanis, Lecture Notes in Computer Science Series (Volume 103). New York: Springer-Verlag.

Lee, E. & MacGregor, J. (1985). Minimizing user search time in menu retrieval systems. *Human Factors*, 27 (2), 157-162.

Lee, E., MacGregor, J., Lam, N., & Chao, G. (1986). Keyword-menu retrieval: an effective alternative to menu indexes. *Ergonomics*, 29(1), 115-130.

Lewis, B. & Crews, A. (1985, March). The evolution of benchmarking as a computer performance evaluation technique. *MIS Quarterly*, 7-16.

Lewis, C. (1986). Understanding what's happening in system interactions. D. A. Norman & S. W. Draper (Eds.), *User centered system design* (pp. 171-185). Hillsdale, NJ: Lawrence Erlbaum Associates.

Lewis, C. & Mack, R. (1981). Learning to use a text processing system: Evidence from "Thinking Aloud" protocols. *Communications of the Association for Computing Machinery*, 387-392.

Lewis, C. & Mack, R. (1982, June). *The role of abduction in learning to use a computer system.* IBM Research Report RC 9433 (#41620).

McDonald, J. E., Stone, J. D., & Lieoelt, L. S. (1983). Searching for items in menus: the effects of organization and type of target. *Proc. of the Human Factors Society -- 27th Annual Meeting,* 834-837.

MacGregor, J. & Lee, E. (1986). Performance and preference in videotext mau retrieval: A review of the empirical literature. *Behaviour and Information Technology.*

MacGregor, J., Lee, E. & Lam, N. (in press). Optimizing the structure of database menu indexes: a decision model of menu search. *Human Factors.*

Mack, R., Lewis, C. & Carroll, J. (1983, July). Learning to use word processors: problems and prospects. *ACM Transactions on Office Information Systems,* 1 (3), 254-271.

Magers, C. (1983). An experimental evaluation of on-line help for non-programmers. *CHI'83 Proceedings,* 277-281.

Marcus, A. (1986, April). User interface screen design and color. *CHI'86 Proceedings* (Boston, MA).

Mark, W. (1986). Knowledge-based interface design. D. A. Norman & S. W. Draper (Eds.), *User centered system design* (pp. 219-238). Hillsdale, NJ: Lawrence Erlbaum Associates.

Marshall, C., Nelson, C., & Gardiner, M. (1987). Design guidelines. In M. M. Gardiner & B. Christie (Eds.), *Applying cognitive psychology to user-interface design* (pp. 221-278). New York: John Wiley & Sons.

Mayer, R. E. (1981, March). The psychology of how novices learn computer programming. *Computing Surveys,* 13 (1), 121-141.

McCann, P. H. (1983). Development of the user-computer interface. *Computers and Education*, 7 (4), 189-196.

McKay, L. (1984). *Soft words, hard words: A common-sense guide to creative documentation*. Culver City, CA: Ashton-Tate.

Melded, Marilyn. (1981). *When People Use Computers: An Approach to Developing an Interface*. Englewood Cliffs, NJ: Prentice-Hall.

Memford, E. (1985, October). *The office of the future: Increasing effectiveness and enhancing the quality of working life.*

Merrill, P. & Tennyson (1978). Step 7: Designing an instruction strategy. *Teaching Concepts: An Instructional Design Guide*, 111-142.

Merrill, P.F. (1982). Displaying text on microcomputers. In D. H. Jonassen (Ed.), *The Technology of Text: Principles for Structuring, Designing, and Displaying Text*. Englewood Cliffs, NJ: Educational Technology Publications.

Miller, L. & Thomas, J. Jr. (1977). Behavioral issues in the use of interactive systems. *Int. J. Man-Machine Studies*, 9, 509-536.

Miyake, N. (1986). Constructive interaction. *Cognitive Science*, 10 (2).

Moran, T. P. (1981). The command language grammar: a representation for the user interface of interactive computer systems. *Int. J. Man-Machine Studies*, 15, 3-50.

Moran, T. P. (1983). Getting into a system: External internal task mapping analysis.. A. Janda (Ed.), *Proceedings of the CHI '83 Conference on Human Factors in Computing Systems* (pp. 45-49). New York: ACM.

Moray, N. (1982). Subjective mental workload. *Human Factors*, 24 (1), 25-40.

Mosier, J. and Smith S. (1986) Application of guidelines for designing user interface software. *Behavior and Information Technology*, 5, 39-46.

Nakatani, L., Egan, D., Ruedisueli, L., Hawley, P. & Lewart, D. (1986, April). TNT: a talking tutor 'N' trainer for teaching the use of interactive computer systems. *CHI'86 Proceedings.*

Neal, A. & Darnell, M. (1984). Text-editing performance with partial-line, partial-page, and full-page displays. *Human Factors,* 26 (4), 431-441.

Neal, A. & Emmons, W. (1984). Error correction during text entry with word-processing systems. *Human Factors,* 26 (4), 443-447.

Norman, D. A. (1984). Studies and levels in human-machine interaction. *Int. J. Man-Machine Studies,* 21, 365-375.

Norman, D. A. (1983). Design principles for human-computer interfaces. *User Centered System Design: Papers for the CHI'83 Conf. on Human Factors in Computer Systems,* ICS Report 8305, 15-24.

Norman, D. A. (1982, May). *Five papers on human-machine interaction.* University of California, San Diego, Center for Human Information Processing.

Osgood, C. (1953). *Method and theory in experimental psychology.* New York, NY: Oxford University Press.

Park, O., Perez, R. S., & Seidel, R. J. (1986). Intelligent CAI: Old wine in new bottles or a new vintage?. In G. P. Kearsley (Ed.), *Artificial Intelligence and Education: Applications and Methods.* Reading, MA: Addison-Wesley.

Parton, D., Huffman, K., Pridgen, P., Norman, K., Shneiderman, B. (1984, June). *Learning a menu selection tree: training methods compared.* U. of Maryland, Center for Automation Research.

Payne, S., & Green, T. (1986). *Task-action grammars.*

Pennington, N. (1985, January). *Stimulus structures and mental representations in expert comprehension of computer programs.* University of Chicago, Graduate School of Business.

Price, Jonathan. (1984). *How to Write a Computer Manual: A Handbook of Software Documentation.* Menlo Park, CA: The Benjamin/Cummings Publishing Company.

Price, Jonathan. (1986). *Help! How to Create Online Help.* Apple Computer, Inc., Cupertino, CA: Alpha Draft.

Price, L.A. (1981). Using Offline Documentation Online. *ACM SIGSOC Bulletin,* 13(2-3), pp. 15-20.

Price, L. A. (1982, March/April). Thumb: An interactive tool for accessing and maintaining text. *IEEE Transactions on Systems, Man, and Cybernetics,* SMC-12 (2), 155-161.

Queipo, L. (1986, December). User expectations of online information. *IEEE Transactions on Professional Communications,* PC-29 (4), 11-15.

Rabinowitz, M. & Glaser, R. (1985, June). *3. Cognitive structure and process in highly competent performance.*

Ramsey, H., Atwood, M. & Willoughby, K. (1979). Paper simulation techniques in user requirements analysis for interactive computer systems. *Proceedings of the Human Factors Society (23rd Annual Meeting -- 1979),* 64-68.

Rehe, R. F. (1984). *Typography: How to make it most legible.* Carmel, Indiana: Design Research International.

Reisner, P. (1981, March). Human factors studies of database query languages: a survey and assessment. *Computing Surveys,* 13 (1), 13-31.

Relles, Nathan. (1979). *The Design and Implementation of User-oriented Systems.* PhD Dissertation, Madison, WI: University of Wisconsin.

Relles, N. & Sondheimer, N. (1981). A unified approach to online assistance. *National Computer Conference, 1981,* 383-388.

Rich, E. (1984, September). Natual-language interfaces. *IEEE Computer,* 39-47.

Rich, E. A. (1982). Programs as data for their help systems. *National Computer Conference,* 1982, 481-485.

Richier, D. and Thompson, K. (1974, July). The UNIX Time-Sharing System. *Communications of the ACM,* 17(7), pp. 365-375.

Riley, M. S. (1986). User understanding. D. A. Norman & S. W. Draper (Eds.), *User centered system design* (pp. 157-169). Hillsdale, NJ: Lawrence Erlbaum Associates.

Rissland, E. L. (1984). Ingredients of intelligent user interfaces. *Int. J. Man-Machine Studies,* 21, 377-388.

Roberts, T. & Moran, T. (1983, April). The evaluation of text editors: methodology and empirical results. *Communications of the ACM,* 26 (4), 265-83.

Robertson, S. (1984, April). *Plans as fundamental units in computer-text editing.* American Educational Research Association (paper prepared for the annual meeting).

Rosenbaum, S. & Walters, D. (1986, December). Audience diversity: a major challenge in computer documentation. *IEEE Transactions on Professional Communications,* PC 29 (4), 48-55.

Rosson, M. B. (1984). Effects of experience on learning, using, and evaluating a text editor. *Human Factors,* 26 (4), 463-475.

Rothenberg, J. (1979). Online tutorials and documentation for the SIGMA message service. *National Computer Conference,* 1979, 863-867.

Rothenberg, J. (1975, May). *An Intelligent Tutor: Online Documentation and Help for a Military Message Service.* Technical Report #ISI/RR-74-26, Univ. of Southern California, Information Sciences Institute.

Rubens, P. (1986). Online information, traditional page design, and reader expectation. *IEEE Transactions on Professional Communications,* PC 29 (4), 75-80.

Rubens, P. & Krull, R. (1985). Application of Research on Document Design to Online Displays. *Technical Communication* 32(4), 29-34.

Rubenstein, R. & Hersh, H. (1984). *The Human Factor: Designing Computer Systems for People.* Burlington, MA: Digital Press.

Scharer, L. L. (1983, July). User training: less is more. *Datamation,* 175-183.

Schell, D. A. (1986). Testing online and print user documentation. *IEEE Transactions on Professional Communications,* PC 29 (4), 87-92.

Schlager, M. & Ogden, W. (1986, April). A cognitive model of database querying. *CHI'86 Proceedings,* 107-113.

Schneider, W. (1984 , August). *Training high performance skills: fallacies and guidelines.* Final report HARL-ONE-8301, Office of Naval Research.

Schultz, E. Jr. & Curran, P. (1986). *Menu structure and arrangement of menu selections: Independent or interactive effects?.* In Jet Propulation Laboratory, California Institute of Technology, Pasadena, CA 91109

Schwartz, J. & Norman, K. (1986). The importance of item distinctiveness on performance using a menu selection system. *Behaviour and Information Technology,* 5 (2), 173-182.

Shannon, R. & Stewart, L. (1986, May). *Human performance aspects of small screen displays: A literature review revealing the lack of specific research.* Technical Report, U.S. Army Human Engineering Laboratory.

Shapiro, S. & Kwasny, S. (1975, August). Interactive consulting via natural language. *Communications of the ACM*, 18 (8), 459-462.

Shneiderman, B. (1986, April). Direct manipulation: an object-oriented visual programming language. *CHI'86 (Human Factors in Computing Systems, Tutorial 17).*

Shneiderman, B. (1986). *Designing the User Interface: Strategies for Effective Human-Computer Interaction*, Reading, MA: Addison-Wesley.

Shrager, J. & Finin, T. *An expert system that volunteers advice.* 339-340.

Schriver, K. A. (1986). *Designing Computer Documentation: A Review of the Relevant Literature -- Hardcopy, Online, General Applications.* CDC TR-31. Pittsburgh, PA: Carnegie Mellon University, Communications Design Center.

Sickler, G. J. (1982, January). System documentation as software. *Conference Proceedings of International Conference on Systems Documentation*, 139-143.

Singley, M. & Anderson, J. *A keystroke analysis of learning and transfer in text editing.* Carnegie Mellon, Deptartment of Psychology.

Smith, D., Irby, C., Kimball, R. & Verplank, B. (1982, April). Designing the Star user interface. *Byte*, 242-282.

Smith, S. and Mosier, J. (1986) *Guidelines for designing user interface software.* (Technical Report MTR-10090). Bedford, MA.: The Mitre Corp.

Smolensky, P., Monty, M. L. & Conway, E. (1984) *Formalizing Task Descriptions for Command Specification and Documentation.* Manuscript submitted to the First IFIP Conference on Humna-Computer Interaction (London, Spetember, 1984).

Soloway, E., Ehrlich, K., Bonar, J., Greenspan, J. (1982, January). *What do novices know about programming?* In Research Report #218, Yale University, Dept. of Computer Science.

Somberg, B., & Picardi, M.C. (1983). Locus of information familiarity effect in the search of computer menus. *Proc. Human Factors Society, 27th Annual Meeting*, 826-830.

Talbott, S. (1983, February). A writer's reflections on product design. *ACM SigDOC Asterisk*, 9 (1), 5-6.

Temin, A. L. (1982, April). *The question answering module in an automated natural language help system for the text-formatter Scribe.* Dissertation proposal, Carnegie Mellon.

Thimbleby, H. (1984). User Interface Design: Generative user-engineering principles. In A. Monk (Ed.), *Fundamentals of Human-Computer Interaction* (pp. 165-180). London: Academic Press.

Tolle, J., Prasse, M. & Gott, R. (1986). Effects of variation in menu length and number of windows on user search time. *CHI'86* (Unpublished manuscript, OCLC Inc).

Twyman, M. (1982). The graphic presentation of language. *Information design journal*, 3/1, 2-22.

Tyler, S., Roth, S. & Post, T. (1981). The acquisition of text editing skills. *Association for Computing Machinery*, 324-325.

U.S. Department of Defense. (1985). *Human Engineering Guidelines for Management Information Systems*, DoD Handbook 761, Washington, D.C.

Vries, J. K. (1985, December). *MARS: an intelligent retrieval system for laser disc medical archives.*

Vygotsky, L.S. *Mind in society: The development of higher psychological processes*. Eds. M. Cole, V. John-Steiner, S. Scribner, & E. Souberman. Cambridge, Mass: Harvard University Press.

Walker, J. H. (1987, December). Issues and Strategies for Online Documentation. *IEEE Transactions on Professional Communication*, PC-30(4), 235-248.

Walker, Jan H. (1986, April). Designing and Implementing Documentation Online: Issues, Strategies, Experience. In *CHI86 Tutorial: Computer and Human Interaction Conference*, Boston, MA, pp. 114-132.

Walker, Jan H. (1984). Symbolics Sage: A Documentation Support System. In *IEEE Spring CompCon '84 Proceedings*, pp. 478-483.

Wasserman, A. (1981). User software engineering and the design of interactive systems. *IEEE*, 387-408.

Weiss, E. H. (1985). *How to write a usable user manual.* Philadelphia: ISI Press.

Weitzman, L. (1986, July). *Designer: A knowledge-based graphic design assistant.* ICS Report 8609, Institute for Cognitive Science, University of California, San Diego.

Welty, C. & Stemple, D. (1981, December). Human factors comparison of a procedural and a nonprocedural query language. *ACM Transactions on Database Systems*, 6 (4), 626-649.

Weyer, S. (1982). The design of a dynamic book for information search. *Int. J. Man-Machine Studies*, 17, 87-107.

Whiteside, J., Jones, S., Levey, P. & Wixon, D. (1985, April). User performance with command, menu, and iconic interfaces. *CHI '86 Proceedings*, 185-191.

Williges, B. & Williges, R. (1981, September). *User considerations in computer-based information systems.* Technical Report, Virginia Tech, Industrial Engineering and Operations Research.

Williges, R. (1984). The tide of computer technology. *Human Factors,* 26 (1), 109-114.

Woods, D. (1984). Visual momentum: A concept to improve the cognitive coupling of person and computer. *Int. J. Man-Machine Studies,* 21, 229-244.

Young, R. M. (1983). Surrogates and mappings: Two kinds of conceptual models for interactive devices. D. Gentner & A. L. Stevens (Eds.), *Mental Models* (pp. 35-52). Hillsdale, NJ: Lawrence Erlbaum Associates.

Yunten, T. & Hartson, H. (1985). A SUPERvisory Methodology And Notation (SUPERMAN) for human-computer system development. In Hartson, H. (Ed.), *Advances in Human Computer Interaction* (pp. 243-281). Norwood, NJ, Ablex Publishing Corp.

Finding Information on a Menu: Linking Menu Organization to the User's Goals

by

Brad Mehlenbacher
Carnegie Mellon
Baker Hall 160
Pittsburgh, PA 15232

Thomas M. Duffy
Indiana University
Instructional Systems
Technology
Education 210
Bloomington, In 47405

Jim Palmer
Carnegie Mellon
and Apple Computer
10500 De Anza Blvd.
MS: 27AR
Cupertino, CA 94051

Prepared for

The United States Army
Human Engineering Laboratory
Aberdeen Proving Grounds, MD

November 20, 1988

This report was prepared for the United States Army Human Engineering Laboratory, Aberdeen Proving Grounds, MD. under contract DAAA1-86-K-0019, contract monitor, Maureen Larkin. The views expressed in this paper are those of the authors and do not necessarily reflect the views of the United States Army, Indiana University, Carnegie Mellon, or Apple Computer

Finding Information on a Menu: Linking Menu Organization to the User's Goals

ABSTRACT

Design paradigms often ignore the diverse goals users bring to the computer interface. Any computer-human interaction can be viewed as a marriage of two systems: the user begins the interaction by formulating an information goal and the computer software meets that goal with a sometimes complex list of potential topic areas. The user then accesses that topic list through the computer interface. Part of the act of accessing the topic list is selecting a potential topic, and this action is often supported by a menu interface. Although research is pervasive on how best to organize menu items to facilitate learning, search speed and reduced selection errors, very little has been done to examine the impact of different types of user goals or cues on a menu's effectiveness. In a study using three distinct cues -- direct match, synonym and iconic -- and two menu organizations -- alphabetical and functional -- data suggests that (1) the functional menu is more effective than the alphabetical menu for the synonym and iconic cues, (2) learning occurs with both menu designs (that is, selection speed increases rapidly across the five trial blocks), and (3) users make fewer errors with the functionally organized menu. The results, in general, encourage more rigorous investigation of the interaction between the tasks users bring to menu interfaces, and the optimal design of those menus.

Finding Information on a Menu: Linking Menu Organization to the User's Goals

1. USER GOALS AND COMPUTER MENUS

Among the various methods designed to facilitate human-computer interaction, menu-driven interfaces have received an immense amount of use and attention over the last ten years (Robertson, McCracken, & Newell, 1981; Card, 1982; Giroux & Belleau, 1986). The addition of menus to existing software, with the supposed guarantee of increased system friendliness, is a standard development in the computer industry; so the essential difference between, for example, CMS and PROFS, Dbase II and Dbase III, PageMaker 2.0 and PageMaker 3.0 (IBM, Ashton-Tate, Aldus Products) is the addition or improvement of a menu -- their functionality is relatively unchanged, only the method of accessing the system's functions has been "enhanced."

The goal in the design of any menu should be to facilitate the user's ability to make a choice quickly and accurately. Human-computer research has tended to focus on two primary issues as most relevant to achieving this goal: the hierarchical organization of menus and the organization of items on any one menu. The major emphasis for hierarchical organization is the proper balance of menu breadth (the number of items one must scan on a given menu) and menu depth (the number of menus one has to pass through to get to the desired information) as they affect search time and selection accuracy.

There are three variables most relevant to determining the proper balance of depth and breadth: the number of items an individual must scan (search time), the number of choices the user must make (decision and motor response time), and system response time to those choices (system time). Adding more menus reduces the number of items a person must scan, but it increases the number of user and system responses that have to be made. For example, with 64 items all on one menu the user, on the average, will scan 32 (half of the) items before arriving at the correct choice (Lee and MacGregor, 1985) but there will only be one user and one system response Alternatively we could create a hierarchical structure of two item

menus so that the user makes a series of six (2^6) binary choices to get to the correct selection. With the six choices, however, the user will (on the average) scan only 1 item per menu for a total of 6 total items scanned to get the correct selection. Calculations of the optimum balance of breadth and depth, based on an assumption of self terminating search (that is, the user quits the search of the menu when the item is located), indicate that the optimum number of items per menu is between 4 and 8 for a wide variety of stimulus sets (Lee and MacGregor, 1985). In general research has supported those calculations (Miller, 1981; Kiger, 1984).

Concern with the breadth-depth issue has tended to dominate the research literature (Allen, 1983; Kiger, 1984; Landauer & Nachbar, 1985). However, recent analyses by Paap and Roske-Hofstrand (1986) suggest that the organization of items on a given menu may be the more important variable for optimizing speed and accuracy of selection. The research on the breadth-depth tradeoff has been based, for the most part, on the assumption that for any one menu, the user begins at the beginning and progresses through it until the needed item is located. That is, the research presumes that the items on the menu are not organized in any meaningful fashion so as to support a systematic search strategy. However, if the items on a menu are organized in some way, and if the user understands or learns the structure of that organization, then the user can skip over groups of alternatives and visually isolate the approximate location of the target item. Given an effective organization, the assumption in the breadth-depth tradeoff studies that the user will on the average scan a minimum of half the items is invalid. With a well defined organization, there need be no difference between a deep menu structure (a few items on many menus) and broad menu structure (many items on a few or even a single menu) in terms of the number of items scanned. Indeed, the deeper structure could even require the user to scan more items than would be required on a single, well structured menu. Thus, the organization of the items on the menu, rather than the depth of the menus, becomes the key determinant of the efficiency of a search.

Certainly, research has shown that the arrangement of menu items effects the amount of time users spend searching for items (Allen, 1983), the number of correct selections made (Lee, MacGregor, Lam, & Chao, 1986), how easily the menu can be learned (Card, 1982), and the overall user satisfaction with the product (Magers, 1983). The issue, however, is identifying the most effective organization of the items for the population of users. The operational definition of "most effective" is

that the organization minimizes search time and selection errors. From a design viewpoint this has two implications. First, the number of groups and the number of items per group must be optimized so as to reduce the number of items a user must search in selecting an answer. This optimization will occur when the number of groups equals the number of items per group; that is, the number of groups equals the square root of the total number of items on the menu.

The second issue, and the issue of concern in the research to be reported here, is the basis on which the items are grouped. Grouping alone may be useful because spacing between groups is a landmark that can be used in maintaining one's place during a search (Rubens and Krull, 1985; Parkinson, Sisson, & Snowberry, 1985). However, the main benefit of grouping will come when the individual can use the structure to guide the search. There are two basic strategies for organizing items on a menu: alphabetical grouping and grouping based on semantic or functional relationships between the menu items.[1] Research has consistently indicated that either of these organizational strategies is better than no systematic arrangement (Card, 1982; McDonald, Stone, & Liebelt, 1983; McDonald and Schvaneveldt, 1988). However, there have been conflicting results when the two approaches have been compared.

Parkinson et al. (1985), for example, failed to find any difference in accuracy or speed when comparing performance with alphabetical or functionally organized menus. In the Parkinson (1985) study, sixty-four items from four naturally occurring groups were presented on a single screen. The organization was manipulated within the four groups of items. The dominant overall organization into four semantic categories may have negated any differential effect of within category differences. However, MacDonald et al. (1983) did find an effect of subgroup organization using much the same stimulus set.

In contrast to these findings, Card (1982) found that selection was faster with an alphabetical arrangement than with a functional arrangement. This study employed a menu of eighteen word processing terms arranged in a single, vertical menu. Subjects were given a command name and asked to select the appropriate

---

1. Actually, there is a third organizational method, one based on an organization within the application. For example, the physical layout of information on a computer console could be the basis for the layout of the menu relating to that information.

target item from the menu. Card's (1982) results have led some authors of interface design texts to include a guideline stating that menus accessing simple reference documentation should be organized alphabetically (Brockmann, 1986, p. 230) and have therefore added credence to the traditional, alphabetical method of organizing menu items.

Finally, in contrast to all of the results presented thus far, McDonald et al. (1983) found that semantic or functional ordering is superior to alphabetical for searching and selecting menu items. MacDonald et al. (1983) presented a sixty-four item menu with items from naturally occurring categories. Subjects were given either a word on the menu (direct match) or a definition of one of the words. Regardless of the stimulus, however, the functional grouping led to the fastest and most accurate selection. As with the Card (1982) findings, this result has also impacted the presentation of development guidelines, except that now we find guidelines recommending a functional rather than an alphabetical organization (see, for example, Shneiderman, 1986, p. 111).

We would argue that the design of a menu must be based on a consideration of the tasks the user must perform in making a menu selection. An analysis of these tasks may help explain the conflicting results and lead to the development of more ecologically valid guidelines. There are two cognitive tasks in making a menu selection. First, the user must represent what information is required. That is, the user must represent the goal in searching the menu. Second, the menu options must be reviewed to determine which one best meets the information needs of the user.

The user's representation of the information need will depend on his or her knowledge base. Here we find it useful to employ Shneiderman's (1986) distinction between semantic and syntactic knowledge. Syntactic knowledge is knowledge of the specific requirements of a particular system; that is, knowing the command names specific to the particular application. Semantic knowledge is broken into two categories: computer concepts and task concepts. Task concepts are those concepts conceptualized apart from the computer, for instance, knowing the writing, editing, and organizational tasks involved in preparing a proposal. Knowledge of computer concepts involves knowledge of the way a task is generally structured in a computer program, for example, the kinds of word processing and file manipulation tasks that

will be involved in preparing the proposal on a computer. It is important to recognize that the user's understanding of the task based on computer concepts could be very different from the understanding of the task based on task concepts (Breuker & de Greef, 1985).

With this distinction in mind, we can now describe three ways in which the user may represent the information he or she is looking for on a menu. First, the user may have syntactic knowledge and therefore know the exact menu item he or she desires. This would be characteristic of most task representations by expert users of the particular application software. Given this goal representation, the user's second task, searching the menu, consists of finding an exact match for the goal representation. Here the perceptual demands are much greater than the cognitive demands of the search.

The second type of representation is based on knowledge of the computer concepts. This is characteristic of a transfer user or an occasional user. That is, the user understands how the task is generally broken down in computer programs and knows the kinds of names attached to the tasks. However, the particular naming conventions in this program are unknown. For example, the user may realize that moving a paragraph generally involves cutting and pasting but he or she does not know whether the relevant commands are "set mark", "cut", et cetera. The cognitive demands of the menu search task are greatly increased given this problem representation. The user must now review each menu item and judge whether or not the command is a synonym for the representation.

Finally, the user may have only task knowledge and not have a representation of the task in terms of computer concepts at all. He or she knows what needs to be accomplished but does not know how to represent the task in computer terms. This type of representation will be common to the novice computer user and may even be common to a user who has some computer experience but who is unfamiliar with a particular class of computer application programs.

In summary, the user may approach the menu looking for a direct match, looking for a synonym for the computer concept, or looking for some computer concepts that seem relevant to the task. If the menu designer is to support the user's search, then the design must accommodate the cue and the knowledge base the user is

bringing to the search task. If the user is experienced in the domain and is looking for a direct match for the stimulus in his or her goal representation, then an alphabetical organization would typically be most effective. That is, an organization based on the alphabetical sequence is extremely well learned and the user should be able to rapidly jump to the correct location on the menu. An organization based on semantic relationships between items may be just as effective as the alphabetical organization if it is as well established, and it may even be more effective if the organizational structure is even more firmly established.

These considerations may help resolve the conflicting results obtained by Card (1982) and MacDonald et al. (1983). Both studies asked users to find a direct match on the menu. Card (1982) found the alphabetical organization most effective while MacDonald et al. (1983) found the functional organization to be most effective. The distinguishing feature is that while Card (1982) employed word processing concepts, MacDonald et al. (1983) used naturally occurring categories of terms. MacDonald et al. (1983) used a card sort strategy to ensure that the organization between categories was maximally distinctive from the relations within categories for the users.

Since the alphabetical structure has been learned and used since childhood in a wide variety of contexts, it is unlikely that any semantic organization of computer concepts will provide as strong a structure for guiding search for a particular, known term. Thus we would expect that the alphabetical organization would provide a more usable structure in the Card (1982) study. However, the semantic relationships used by MacDonald et al. (1983) were learned since childhood and are used in even a wider variety of circumstances than the alphabetical sequence. It is reasonable to assume, therefore, that the semantic organization could provide a more supportive structure than the alphabetical organization -- or at least that results would be inconsistent when two well learned structures are compared.

Most research has focused on the direct match search task. MacDonald et al. (1983), however, also used the definition of words in the menu as the cue. Thus semantic, not just syntactic, knowledge was required. As we would predict, the functional organization was even more effective, relative to the alphabetical organization, in supporting this task.

The focus of the present research is on the design of menus for computer systems, that is, for menus where the semantics are not overlearned. MacDonald et al. (1983) found that for videotex-based menus, a semantic organization was most effective regardless of the user's task. We would predict that with computer menus, the most effective menu organization will depend on the user's task or problem representation. For a direct match, characteristic of the experienced user, an alphabetical organization should be most effective. However, when the user only has semantic knowledge, a functional or semantic organization will be most effective provided, of course, that it reasonably reflects the user's semantic organization.

The present research compared the effectiveness of two types of command-based menu organizations in supporting the efficient location of the appropriate command given one of three kinds of cues: an exact command to be located on the menu, a synonym for the menu item, or an iconic representation of the action performed by the command. We used a menu containing word processing and formatting commands and tested users who had some computer and word processing experience. The users, therefore, were equivalent to transfer users with a low level of general experience. The iconic representation involved the presentation of before and after texts where the before showed graphically (without words) the kind of change to be made and the after showed the text with the changes completed. We would expect that the users, given their level of experience, will have greater difficulty matching a menu item to the iconic stimulus than to the synonym.

2. EXPERIMENTAL METHOD

2.1. Design and Materials

Three experimental variables were manipulated in a 2 x 3 x 5 mixed factorial design. The first variable, Menu Organization, consisted of an alphabetically organized menu (see Figure 1) and a functionally organized menu (see Figure 2). Both menus had twenty-two items separated into three columns. The number of menu items approximates many of the conventional word processing menus with between fifteen and twenty-five items to choose from (see, for example, WordStar, MicroSoft

Word, EZ). For the functional menu, a category heading in bold typeface was over each group of items.

INSERT FIGURES 1 & 2 HERE

The second variable, Cue, consisted of three types of cues meant to capture the different types of goal tasks users bring to a word processing menu:

- a direct match cue (that is, "I know the word that the application uses and I want to find it"). For the direct match cue the subject would be shown, for example, the word "Center" and asked to select the command "Center" on the menu.

- a synonym cue (that is, "I know what I want to do, but do not know the exact command the application uses"). For the synonym cue the subject would be given the phrase "Move text to middle" and asked to select the command "Center" on the menu.

- an iconic cue (that is, "I have a picture of what I want the document to look like -- before and after -- and I want to find the command that will allow me to accomplish it"). In the iconic cue the subject would be shown a picture of a text before and after a change had been made to it (see Figure 3).

INSERT FIGURE 3 HERE

2.3. Subjects

Seventy-five university students acted as subjects for the experiment. Subjects were screened before-hand, and only individuals who had some computer word processing experience were permitted to participate. Before they began the experiment, they were asked to fill out a short questionnaire asking for basic demographic information (year in college, major or program name, et cetera) and

word processing background information (years of experience, how many times they had used a word processor in the last two weeks, which word processor or processors had they used). Subjects were paid two dollars per session or received course credit for their participation.

2.4. Procedure

Eight groups of subjects were tested with five to twelve subjects per group. When subjects entered the testing room they were handed a diskette and asked to sit at one of the microcomputers. Each diskette contained all of the instructions and the test administration for one of the experimental conditions. The disks were distributed randomly within sets of six disks representing the six conditions. All conditions were therefore represented approximately equally within each group that was tested.

Subjects were told that we were evaluating alternative designs for menus and that they would be working with word processing terminology. They then placed their diskette in the machine and individually progressed through the condition-specific instructions and testing. The instructions for the experiment were presented on two pages of the computer screen. Across the three cues, instructions varied slightly in wording but not in presentation. They were told that they would be shown a word processing term or text with editing marks and then that same text with the editing completed. Their task was to find the word on a menu that was the same as (or that described) the cue; this process, they were told, would be repeated for some time. The instructions stressed the importance of selecting the correct term on a menu, as well as the importance of making selections as quickly as possible.

The subjects received five practice trials using animal names (and pictures for the iconic condition) after reading the instructions. The trials in practice and in the main experiment progressed as follows. A ready signal was presented for two seconds and was then replaced with the cue. The subject studied the cue and then clicked the mouse in a box at the bottom of the screen when he or she was ready to see the menu. When the menu appeared, the subject's task was to click on the correct menu item for the cue. No feedback on the correctness of the choice was provided. After the selection, the menu was replaced with the ready signal to mark the beginning of the next trial.

After the practice, subjects were reminded of the instructions and told they would have one minute to study the word processing menu before testing again. They were asked to familiarize themselves with the word processing terminology and the menu's content and format.

After one minute, the menu was removed and the testing began. There were five blocks of fourteen trials. The same fourteen commands were tested in each block of trials and for all conditions. The fourteen commands (out of the twenty-two possible menu items) were selected so that they were spread across each of the three columns in the alphabetical menu and each of the six functional groups in the functional menu. The commands, which were randomly ordered for each of the five trial blocks, were Bold, Center, Copy, Delete, Exit, Find, Insert, Italic, Justify, Move, Print, Underline, Upper and Windows (see Figures 1 and 2).

3. RESULTS AND DISCUSSION

Student responses to the preliminary questionnaire indicated that their experience with word processors ranged from six months to five years (X = 2.3, SD = 1.3). The types of word processors used by the subjects varied and included AppleWriter, Ed, EMACS, Final Word, MacWrite, MicroSoft Word, WordStar, and Xedit. The average number of reported uses during the last two weeks was 5.8 (SD = 4.8).

3.1. Problem Formulation

When a user goes to a menu to obtain information or to execute a command, he or she must first formulate the information need in terms of what he or she expects to find on the menu. In short, the user must translate the need into an appropriate menu selection goal. The time spent studying the cue, then, may be taken as an indicant of the user's effort in this problem formulation phase. That is, prior to requesting the menu, the participant was permitted to study the cue as long as he or she desired. Then, once the menu was requested, a selection was to be made as fast as possible.

Since the same cues were used in each block of trials, we obviously anticipate that this study time (or problem formulation phase) will decrease significantly across trial blocks. Furthermore, the direct match cue should require the least study time

since the participant simply must find that cue on the menu. The iconic cue should require the greatest study time since the participant must first study the change and attach a name to it. Then he or she must evaluate the appropriateness of the name in terms of the word processing terminology in general and of the menu in particular.

A 2 by 3 by 5 mixed factorial ANOVA was performed on the mean study time. The between subjects variables were Cue (direct match, synonym or iconic) and Menu Organization (alphabetical or functional). The within subjects factor was Trial Block (five blocks of 14 trials). The results were as anticipated: the main effects of Cue and Trial Block were significant, $F(2,68)= 14.4$ and $F(4,272)= 112.1$, respectively, $p<.001$, as was the interaction of Cue and Trial Block ($F(8,272)= 44.13$, $p<.03$). However, there was also an unanticipated main effect of Menu ($F(1,68)= 3.97$, $p<.05$) as well as a significant three way interaction ($F(8,272)= 2.29$, $p<.03$).

The Cue by Trial Block interaction is illustrated in Figure 4 where it can be seen that study time for the iconic cue was more than twice as long as the study time for the other cues, while the study time for the direct match cue is somewhat less than for the synonym cue. However, the effects diminish rapidly across trials such that there is no difference between the cue conditions by the third trial block.

INSERT FIGURE 4 HERE

The main effect of Menu Organization was due to a somewhat fast study time in the functional menu condition. While the difference persisted across trial blocks, the primary difference was in the first trial block where the mean study times were .448 and .371 s for the alphabetical and functional menu conditions respectively. Furthermore, the advantage of the functional menu was greatest for the iconic cue condition ($\bar{X}$ = 809 and 560 s in the first trial block for the alphabetical and functional menus respectively) but was very minor with the direct match cue (a difference of .006 s in the first trial block). This effect reflects the significant three way interaction.

It is surprising that people making selections from the functional menu required less study time than people selecting from the alphabetic menu. The menu is not presented until after the study time is terminated by the subject and both the functional and the alphabetic menus contain exactly the same terms (with the exception of group labels in the functional menu). The subject, therefore, must somehow use their memory of the menu's structure to guide them in interpreting the cue. While not anticipated, the effect of menu structure on cue study time can be interpreted within our conceptual framework. That is, we would contend that the functional organization provides a conceptual structure for the set of commands – a schema for thinking about the problem domain. Not only should the schema facilitate search but it should also assist the user in formulating his or her information needs. In this respect, the functional organization is a structure that can be used to guide problem formulation.

There are two possible explanations for why the functional organization facilitates the problem formulation phase. First, the functional organization provides a structure that facilitates the user's ability to learn the command names on the menu. Items in semantic categories are easier to learn than uncategorized items. As a result of this learning, the user should possess a greater repertoire of commands on which to base his or her information needs and ultimate decision. Second, the functional menu provides conceptual bins or chunks that help the user classify the cue or at least identify the general location of the appropriate command. Both of these interpretations would predict, as we have seen, that the functional menu would have a significant effect on the behavior of the user who studied the iconic cues and virtually no effect on the user who studied the direct match cues.

3.2. Menu Selection Time

Once the participant formulated the problem, he or she requested the menu and made a menu selection as rapidly as possible. We submitted the menu selection time to the 2 by 3 by 5 mixed factorial ANOVA where the between subjects variables were Menu Organization and Cue and the within subjects factor was Trial Block. The only main effect to achieve significance was Trial Block ($F(4,272)= 49.6$, $p<.001$) and this simply reflected the fact that the performance of all the groups improved across trial blocks. The effects of Menu Organization and Cue Type appear in the interaction with Trial Blocks. The Menu by Trials interaction yielded an $F(4,272)=$

2.51, $p<.05$, while the Cue by Trials interaction yielded an $F(8,272)= 4.39$, $p<.001$. The three way interaction also reached significance ($F(8,272)= 2.3$, $p<.02$).

The interaction of Menu Organization and Trial Blocks is illustrated in Figures 5, 6, and 7 for the direct match, synonym, and iconic cues respectively. As can be seen in the figures, the groups all come together in the later trial blocks, with mean performance ranging between .220 and .173 s. A post hoc comparison using the Duncan Multiple Range test failed to yield any significant differences between the six groups on this last trial block. The primary effects are seen in the early trial blocks.

INSERT FIGURES 5, 6 & 7 HERE

Looking at the first trial block we see that the alphabetic menu leads to faster selection time than the functional menu only under the direct match condition, that is, when the cue is an item from the menu. As we expected, the participants are well versed in the alphabetical structure and the ability to locate a command based on an alphabetical organization transfers easily to this task. If the functional structure of word processing terms was also well known we would expect the functional organization to be equally effective.

The opposite effect occurs when we look at the synonym and iconic conditions. In both cases, the functional organization leads to faster menu selection time. The difference between the functional and alphabetical menus is very large for the iconic condition (a difference of almost 3 s in the first trial block) and the effect persists across all blocks.

3.3. Selection Errors

The participants in the direct match conditions made virtually no errors ($X = 0.04$) in selecting menu items. Since they were simply selecting the exact item presented, no errors were expected. However, participants in the synonym and iconic conditions had to select the menu item they felt would accomplish the task represented or which meant the same as the synonym presented. There were errors in these

conditions and, as one would expect, there were more errors with the iconic cues than with the synonyms.

Of interest is whether or not the functional organization not only led to faster selection time but also to more accurate selections. We would anticipate that the shared meaning achieved by a functional grouping would help the participant infer the meaning of a command and therefore make a more accurate linkage between the task (represented by the cue) and the command.

Mean errors per Trial Block are presented in Figure 8 for the synonym and iconic Cue conditions. As can be seen, the functional menu leads to a lower error rate with both Cue Types. In addition, while error rate decreases across trial blocks, participants still made fewer errors with the functional menu even on the last trial block.

INSERT FIGURE 8 HERE

The error rate for the Synonym condition was too low, with too many zero scores, to submit to an analysis of variance. However, the data from the iconic Cue condition was analyzed in a 2 by 5 mixed factorial ANOVA with Menu Organization and Trial Blocks as the factors. Trial Blocks, as expected, yielded a significant effect, $F(4,92)= 8.92$, $p<.001$. Similarly, the main effect of Menu Organization was significance ($F(1,23)= 4.78$, $p<.05$). The interaction of Menu with Trial blocks, however, did not reach significance. The results confirm the relations apparent in Figure 8.

It is possible that this difference in errors may account for the difference in selection time. That is, it may be that the participants take longer when making error responses and, since they made more errors with the alphabetical menu, mean selection time will also be longer. We have assumed that the functional organization facilitates search: the user can look at a category title or an exemplar of the category and quickly judge whether or not it is a relevant group to search. However, if the selection time effect is restricted to error responses, we must reject this hypothesis and conclude that the effect of the functional organization is due to the error reduction that occurs through specific semantic effects; by combining

correct and incorrect selection times, we may be looking at decision making rather than actual search time.

To test this hypothesis, we re-analyzed the time required to make the menu selection, but this time, we only included the time data for correct selections. Since the majority of errors were in the iconic condition, we only conducted this re-analysis for the iconic Cue conditions. A 2 by 5 ANOVA of the mean time per trial block for correct choices yielded a significant main effect for Trial Blocks ($F(4,92)= 25.02$, $p<.001$) and a significant Menu by Trial Block interaction ($F(4,92)= 3.28$, $p<.05$). The results illustrated in Figure 9 parallel the time data for all responses, confirming that the time effects are not due to longer times on error responses.

INSERT FIGURE 9 HERE

---

4. CONCLUSION

4.1. Implications for Interface Design

The results confirm our hypothesis that the most effective or the "best" organization of a menu depends on the user's knowledge base and information goal. For the individual who knows what command he or she wants, the alphabetic structure will typically be most effective. If the alphabetical structure of information is the strongest cognitive organization of the set of commands, then it will be most effective. This was the case for our menu and we may expect it to be the case for most menus from application programs. If the user "knows" the knowledge domain very well -- has a long history of experience with the item set -- then we may expect his or her functional model of the organization of items in the domain will be equivalent to an alphabetical organization. Hence, alphabetical and functional organizations would be equally effective in a direct match task (presuming of course that the appropriate functional organization is presented, Palmer et al., 1987). This could be the case for the well practiced expert using an application program. It would also be the case for any user searching a menu in a retrieval system dealing with common information (for example, Hollands & Merikle, 1987).

More typically, however, users will not be well practiced nor will they have an overlearned organization of the knowledge domain (as would be found with popular databases). For this more typical case, a functional organization will be more effective when the user does not know exactly what command is required – as is often the case with novice and intermediate users, . Indeed, our results suggest that the greater the user's uncertainty about the desired command, the more effective the functional organization will be. In essence, we are suggesting the following: first, that menu organization will make little difference for the expert (who knows the command he or she wants and has a very strong functional organization of the knowledge domain); second, that an alphabetic organization will be most effective for a wider range of experts (who generally know the command they want), and; third, that a functional organization will be most effective for novices (who do not know what command they need). However, casting this as an expertise issue is really a means of summarizing the more precise characterization in terms of user knowledge and task. Therefore, the relation to expertise outlined above is made with two provisos.

First, user expertise refers to the particular command set -- expertise is not a general characteristic, even in terms of particular application software. For example, even "experts" of complex word processors like Xedit or EMACS typically do not use a large set of the program's commands. There are subsets of EMACS commands or functions with which the EMACS expert may well be a novice. As long as the expert restricts the use of an application like EMACS to the familiar commands or functions, the alphabetical organization should be as effective or more effective than the functional organization. However, when these experts begin to extend their use of Xedit or EMACS still further and begin using new commands, then we would predict a functional menu would be most effective.

The second proviso is that expertise is not developed as rapidly as one might think by looking at performance across Trial Blocks. Performance for our groups comes together quite rapidly such that there are few differences after just seventy trials. This might lead one to believe that menu organization is not an important issue since it does not make much of a difference after the first five minutes of experience with the program. Of course the experimental context provides a very compressed experience within a very limited context. That is, our users worked continuously at

the task, selecting one command after another. This is quite different from the normal work experience where many commands would not get used at all during an hour long session and where the user may only use the program once a week or once a month. Also, in the normal work environment the cues or context for a particular command are continually changing. We used consistent stimuli -- the same icons -- from trial to trial but, of course, the user of computer software is always facing new stimuli and new problems.

MENU

Bold	Get	Print
Cancel	Insert	Put
Center	Italic	Replace
Copy	Jump	Underline
Delete	Justify	Undo
Exit	Margin	Upper
Font	Move	Windows
Find		

Click the mouse over the correct menu item.

Figure 1. The Alphabetically Organized Menu.

MENU

Changing Text	**Formatting**	**Control Commands**
Insert	Bold	Windows
Delete	Italic	Undo
Replace	Underline	Cancel
Move	Upper	Print
Copy	Font	
Finding Text	**Alignment**	**Files**
Find	Center	Get
Jump	Margin	Put
	Justify	Exit

Click the mouse over the correct menu item.

Figure 2. The Functionally Organized Menu.

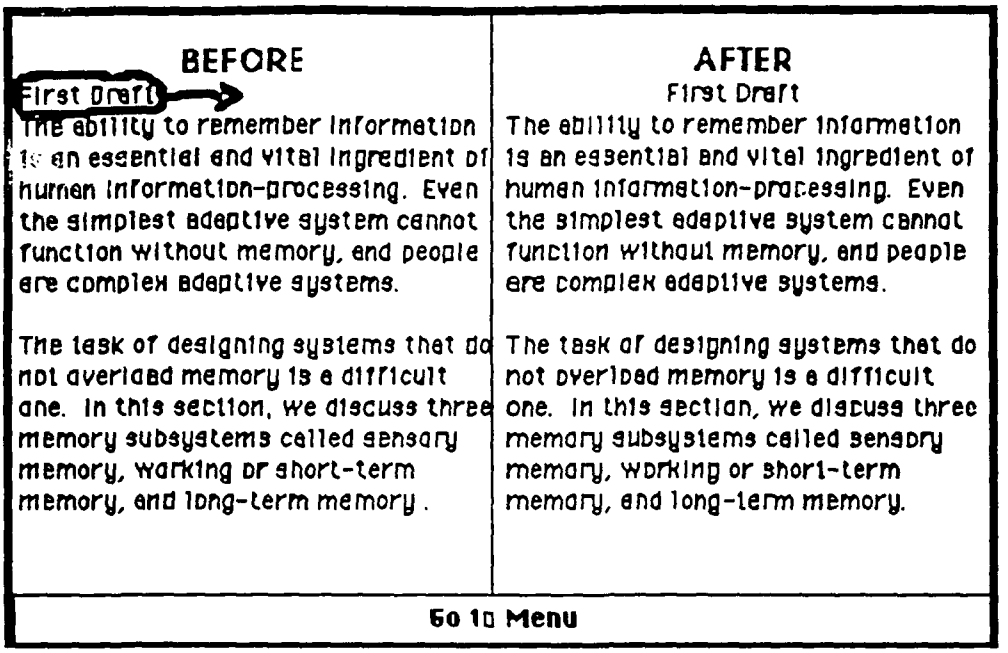


Figure 3. Iconic Cue representing "Center".

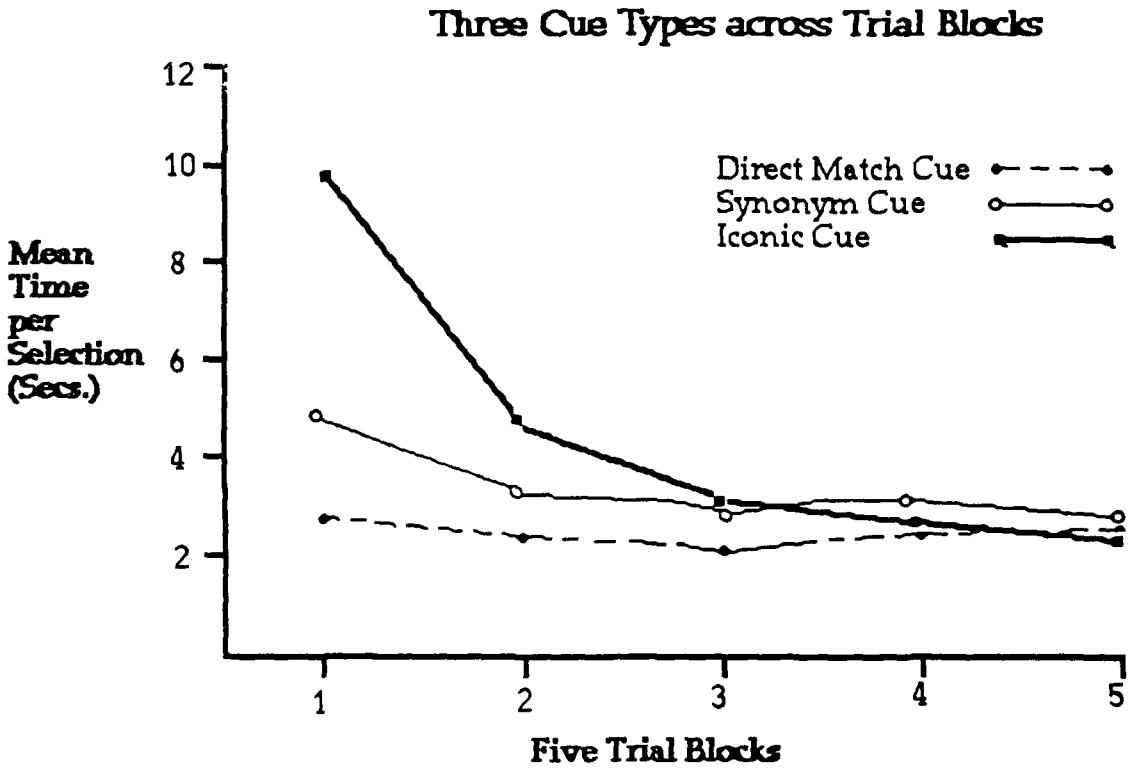


Figure 4. Study time for direct match, synonym and iconic Cue types across the five Trial Blocks.

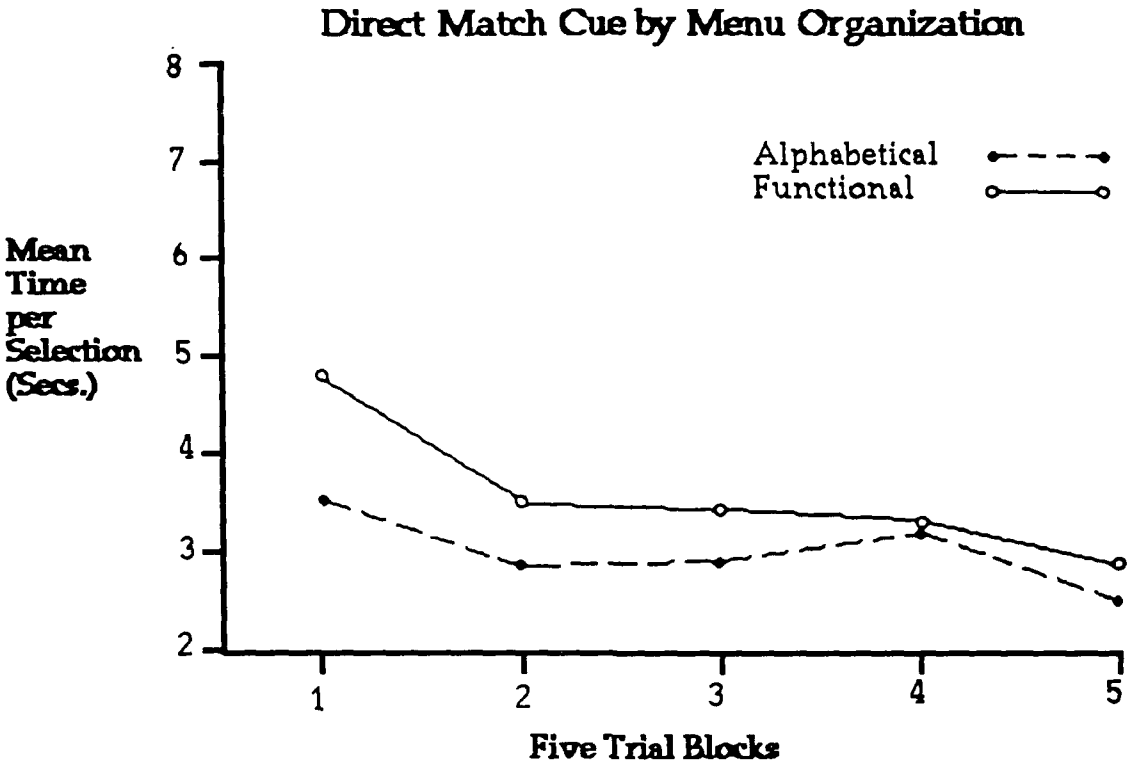


Figure 5. Mean menu selection time for the alphabetical and the functional menus in the direct match cue condition across the five trial blocks.

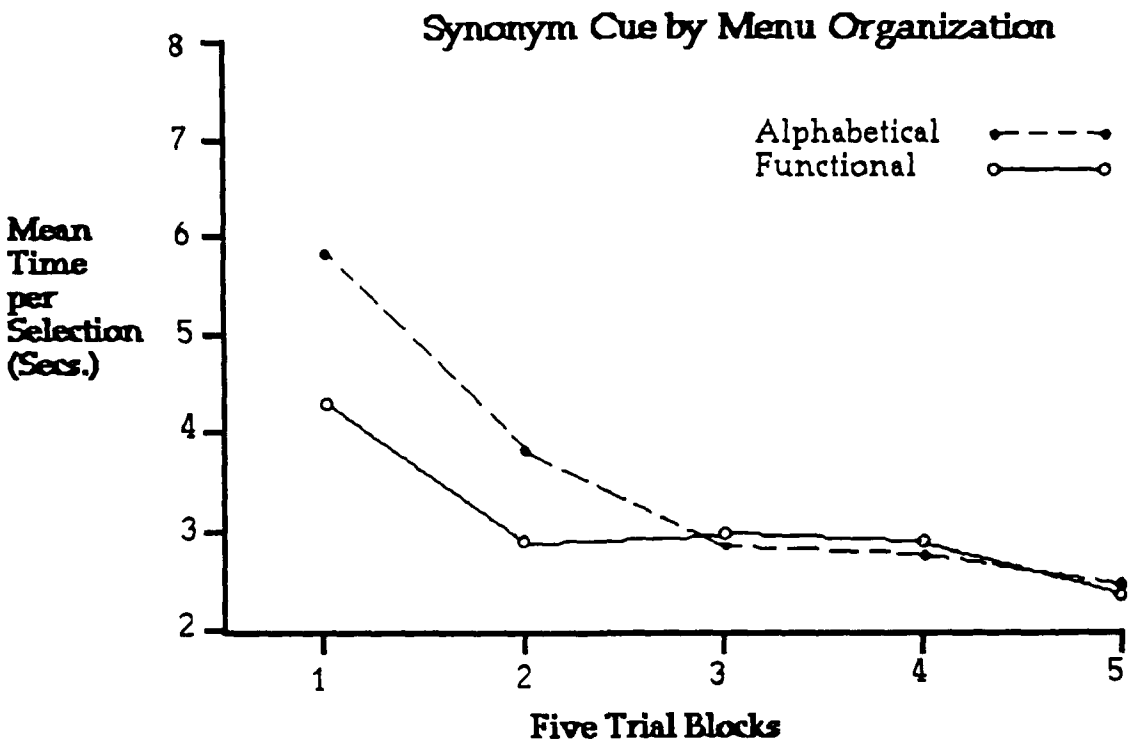


Figure 6. Mean menu selection time for the alphabetical and the functional menus in the synonym cue condition across the five trial blocks.

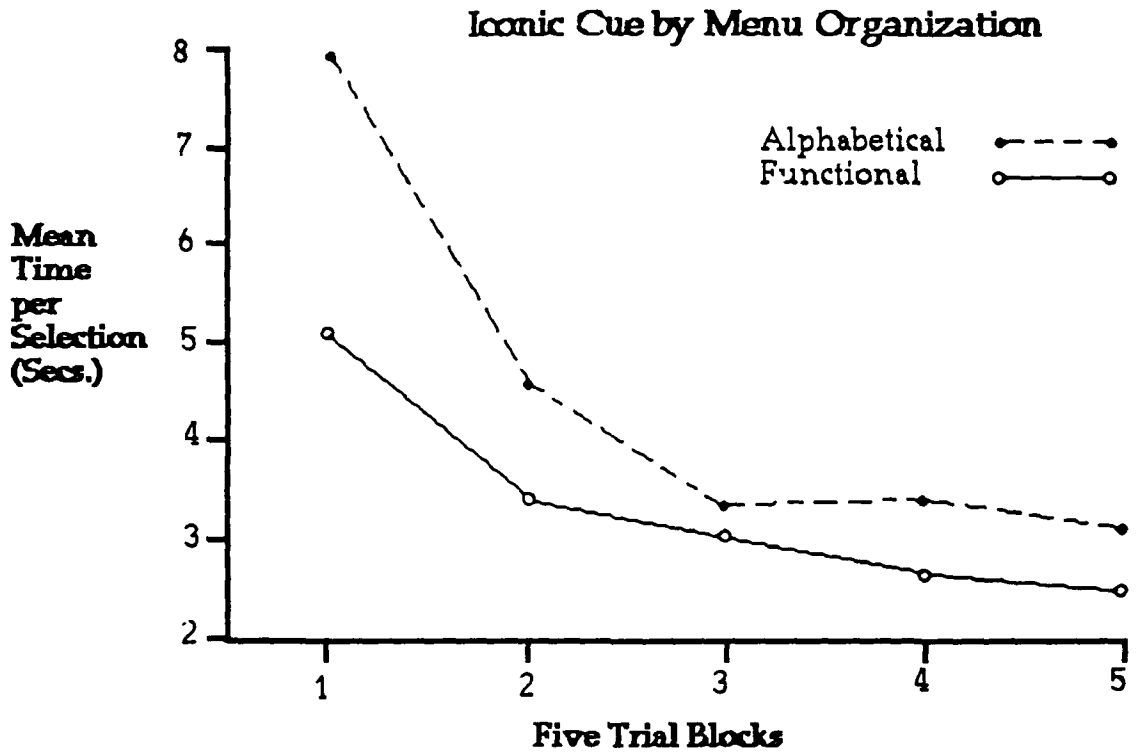


Figure 7. Mean menu selection time for the alphabetical and the functional menus in the iconic cue condition across the five trial blocks.

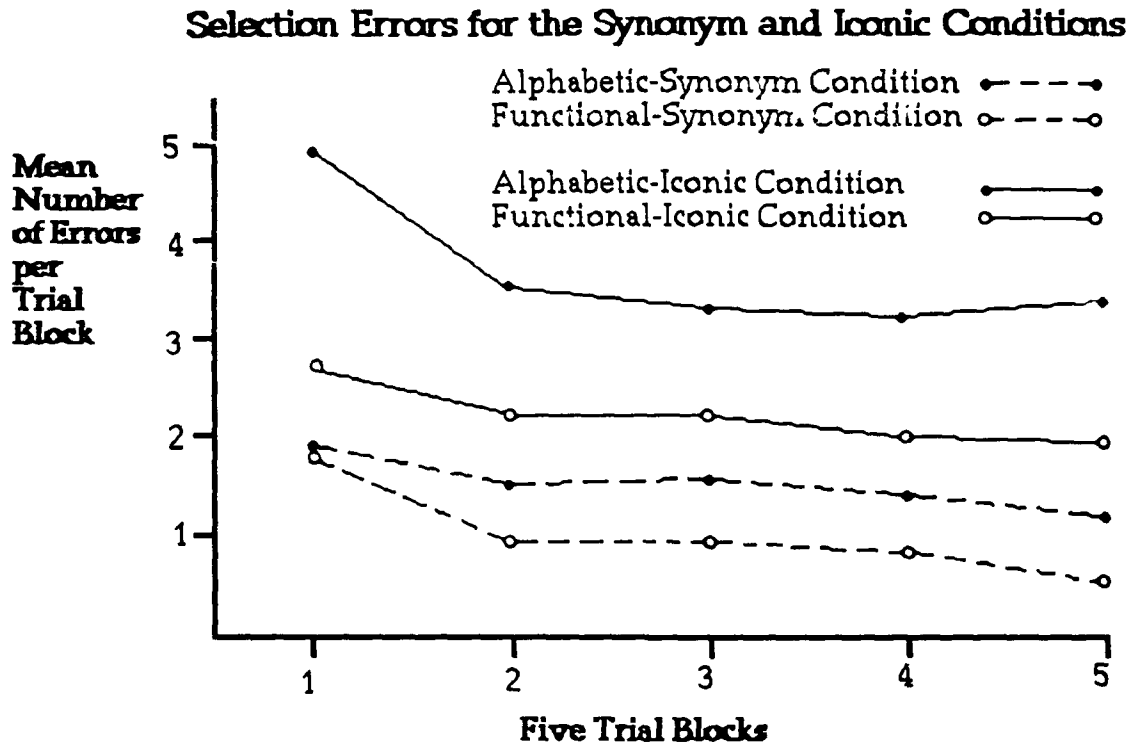


Figure 8. Mean number of selection errors for the alphabetical and the functional menus in the synonym and iconic cue conditions across the five trial blocks.

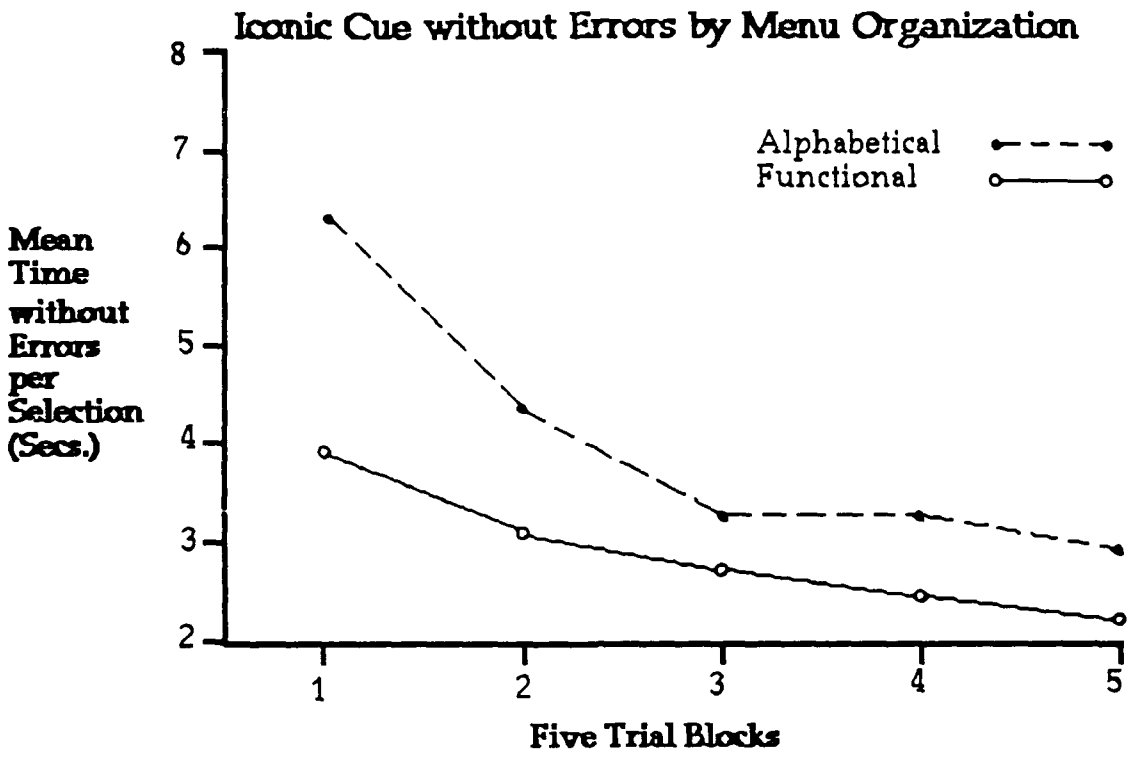


Figure 9. Mean number of selection without errors for the alphabetical and the functional menus in the iconic cue condition across the five trial blocks.

References

Allen, R. B. (1983). Cognitive factors in the use of menus and trees: an experiment. *IEEE Journal on Selected Areas in Communications, SAC-1(2),* 333-336.

Brockmann, R. J. (1986). *Writing Better Computer User Documentation: From Paper to Online.* NY: John Wiley & Sons.

Breuker, J., & de Greef, P. (1985). Information Processing Systems and Teaching & Coaching in HELP Systems. *Department of Social Science Informatics, University of Amsterdam,* Herengracht 196, 1016 BS Amsterdam.

Card, S. K. (1982). User Perceptual Mechanisms in the Search of Computer Command Menus. *Proceedings of the CHI '82 Conference on Human Factors in Computer Science,* 190-196.

Giroux, L., & Belleau, R. (1986). What's on the menu? The influence of menu content on the selection process. *Behaviour and Information Technology, 5(2),* 169-172.

Hollands, J. G., & Merikle, P. M. (1987). Menu organization and user expertise in information search tasks. *Human Factors, 29(5),* 577-587.

Kiger, J. L. (1984). The depth/breadth trade-off in the design of menu-driven user interfaces. *Int. J. Man-Machine Studies, 20,* 201-213.

Landauer, T. K., & Nachbar, D. W. (1985). Selection from alphabetic and numeric menu trees using a touch screen: breadth, depth, and width. *Proc. Human Factors in Computing Systems, ACM SIGCHI,* 73-78.

Lee, E., & MacGregor, J. (1985). Minimizing User Search Time in Menu Retrieval Systems. *Human Factors, 27(2),* 157-162.

Lee, E., MacGregor, J., Lam, N., & Chao, G. (1986). Keyword-menu retrieval: an effective alternative to menu indexes. *Ergonomics, 29(1),* 115-130.

Magers, C. (1983). An experimental evaluation of on-line help for non-programmers. *Proceedings of the CHI '83 Conference on Human Factors in Computer Science,* 277-281.

McDonald, J. E., Stone, J. D., & Liebelt, L. S. (1983). Searching for items in menus: the effects of organization and type of target. *Proceedings of the Human Factors Society, 27th Annual Meeting,* 834-837.

McDonald, J. E., & Schvaneveldt, R.W. (1988). The application of user knowledge to interface design. *Cognitive Science And Its Applications For Human-Computer Interaction.* Guindon, R. (Ed.). NJ: Lawrence Erlbaum Associates. 289-338.

Miller, D.P. (1981). The Depth/Breadth Tradeoff in Hierarchical Computer Menus. *Proceedings of the 25th Annual Meeting of the Human Factors Society,* 296-300.

Paap, K. R., & Roske-Hofstrand, R.J. (1986). The optimal number of menu options per panel. Human Factors. *Human Factors, 28(4),* 377-385.

Parkinson, S. R., Sisson, N., & Snowberry, K. (1985). Organization of Broad Computer Menu Displays. *Int. J. Man-Machine Studies, 23,* 689-697.

Robertson, G., McCracken, D., & Newell, A. (1981). The ZOG approach to man-machine communication. *Int. J. Man-Machine Studies, 14,* 461-488.

Rubens, P., & Krull, R. (1985). Application of Research on Document Design to Online Displays. *Technical Communication,* Fourth Quarter.

Shneiderman, B. (1986). *Designing the User Interface: Strategies for Effective Human-Computer Interaction,* Reading, MA: Addison-Wesley.

The Evaluation of Online Help Systems:

A conceptual model

A Technical Report

Prepared by

Thomas M. Duffy
Indiana University
Instructional Systems
Technology
210 Education
Bloomington, In 47405

Brad Mehlenbacher
Carnegie Mellon
Baker Hall 160
Pittsburgh, PA 15232

Jim Palmer
Carnegie Mellon
and Apple Computer
10500 De Anza Blvd.
MS: 27AR

for

The United States Army
Human Engineering Laboratory
Aberdeen Proving Grounds, MD.

30 November 1988

This report was prepared under contract DAA1-86-K-0019 for the U.S. Army Engineering Laboratory, Aberdeen Proving Grounds, MD. Ms. M. Larkin was the technical monitor for the contract. The views expressed in this paper are those of the authors and do not necessarily reflect the views of the U.S. Army, Indiana University, Carnegie Mellon, or Apple Computer.

The Evaluation of Online Help Systems: A conceptual model

Abstract

There are an increasing number of software applications providing users with assistance online. We might even anticipate that online assistance will be the standard means of aiding a user in the near future. This chapter examines one online strategy for aiding a user: online help. Help systems are distinguished from other types of assistance and a conceptual model for analyzing online help systems is presented. Finally efforts to use that model in designing an online help evaluation system are described.

The Evaluation of Online Help Systems: A conceptual model

1. Introduction

> *Technology, properly interiorized, does not degrade human life but on the contrary enhances it.*
>
> Walter Ong

New technology, as Walter Ong notes in *Orality and Literacy*, has the power to *restructure* human consciousness: to transform the ways we interact with and think about the world. Writing itself is a technology: it requires a whole array of tools and "equipment." Moreover, the technology of writing has transformed our way of viewing the world. It allows us to form grapholects such as standard English (with nearly 2 million words) and to develop fully analytic and abstract modes of thinking. We fail to regard writing as technology because we have interiorized it (Ong, 1982).

The computer extends the technology of writing to a new medium. For example, accountants who once balanced books using ledgers and adding machines can now interact with electronic spreadsheets. As a new tool, the spreadsheet requires the accountants to perform their original tasks in new ways; ultimately, it transforms the way in which they understand those tasks.

To use the computer effectively, users must "properly interiorize" its technology, much as a concert pianist must interiorize the technology of the piano and a musical score. We claim that **helping computer users to interiorize technology** is the ultimate goal of training and assistance (such as tutorials and documentation). Training and assistance should help users to

1. establish a context for using an application,
2. learn how to achieve their goals using that application, and
3. acquire the skills necessary for performance.

We also believe that users will always require some training and assistance, no matter how simple or obvious a computer design aims to be (cf. Carroll & Mazur, 1985).

Delivering training and assistance online seems to be a natural way to use the flexibility provided by the computer. In theory, at least, presenting documentation and tutorials online has several advantages over the print medium (Brockmann, 1986; Shneiderman, 1987; Walker, 1987; Duffy, Gomoll, Gomoll, Palmer, & Aaron, 1988). Online delivery allows

Greater availability — With networks and portable computers becoming more prevalent, online information can provide a reliable source of information as hardcopy documentation becomes less available or less convenient to access.

Easier access — Online, the system can provide mechanisms for efficient access to the relevant information, especially in cases where that information might span many volumes of hardcopy documentation.

More interaction — Online, both the user and the system can interact with the information. For example, the system can use the state of the application (its current context) to determine what information to provide to the user, or a monitor capable of plan recognition could help debug users' faulty or inefficient procedures.

Low cost/High accuracy — In general, online information is less expensive to store, produce, and update. Hardcopy documents require much longer production cycles. As computer companies adopt shorter and more efficient software development cycles, the pressure to adequately document a product increases. The time it takes to produce a book after it is written—layout, formatting, and printing—becomes a bottleneck. Either manuals go into production well before products are stable (resulting in manuals that are inaccurate or incomplete) or a company incurs costs in order to make the necessary changes and to begin the production cycle again.

Multimedia and AI — Online information can exploit multiple media, such as video, sound, and animation, and can apply techniques from Artificial Intelligence (AI).

Of course, there are disadvantages to the online medium as well. First, with most computer systems users cannot work in the primary application while using the online information. In contrast, users can work comfortably with an application while using a hardcopy tutorial or manual. Second, it is well

documented that most computer displays diminish the readability of text and the legibility of characters, two factors which make reading from screens more difficult than reading from a book or manual (Muter, Latremouille, Treurniet, & Beam, 1982; Kruk & Muter, 1984; Haas & Hayes, 1987). In general, these types of problems are *technological*: advances in computer hardware and software may reduce or eliminate the difficulties. There also remain *conceptual* problems. For example, the familiar strategies for navigating through books do not apply to online information (cf. Robertson & Akscyn, 1982; Elm & Woods, 1985). Users, therefore, must learn how to interact with the new medium.

Our goal is to address some of the problems posed by online training and assistance. In particular, we want to provide a conceptual model for understanding online help systems. We derive our model from an analysis of the cognitive tasks required of the user of the online help system. From this model we develop several methods for evaluating existing online help systems. The same model can also be used to guide the design of help systems, although we do not focus on design issues in this paper. To present our conceptual model for evaluating online help, we will

- more formally define online help;
- discuss the goals and methods of evaluation; and
- present our method of evaluation based on a cognitive task analysis.

2. What is online help?

Unfortunately, the term online help carries a number of competing connotations. One way to define online help is to classify various implementations of online information along some scale. For example, Shirk (1988) creates a taxonomy of online information based on two scales: increased writing complexity and progressive self-containment. At the low end of the scale she places system messages, at the high end, computer-based training. In contrast, we follow an approach based on Kinneavy's *Theory of Discourse* (1972) and reframe the question from one of classifying products (various implementations of online information) to one of classifying the aims of users in specific situations or contexts: what do they want or expect from online information? What questions should online systems be able to answer? Note that we view the function of information with respect to the context of its use, not the writer's intentions or the static description of its form (cf. Bethke, Dean, Kaiser, Ort, & Pessin, 1981).

The core idea behind our effort is to match the information provided to users with the different kinds of knowledge that they require. This focus allows us to establish general distinctions that do not rely on the specifics of an implementation (for example, does the system have a mouse?). Actual help systems, then, are a cross between content and the constraints imposed by online presentation. These constraints obviously include different hardware and software platforms, as well as available resources and creative differences among designers. But whether one is designing help for an IBM-PC or a Unix-based workstation, our goal is to match content to the user's goals and to present that content effectively.

Two basic types of user-accessed online information[1]

1. *Primary sources (Browsing).* The exemplars for this category include an online encyclopedia or a bibliographic system for retrieving articles about various subjects. The **reference** of this discourse is primary in that the information supports the solution of a real-world problem, such as writing an article on electricity or finding the current stock prices in the Wallstreet Journal. Its **audience** is all users, and their **goal** is to browse or look up information relevant to their primary task.

2. *Secondary sources (Understanding).* The exemplars for this category include a wide variety of forms considered "electronic documentation," which we discuss in more detail below. The **reference** of this discourse is secondary, in that the information supports the use of a tool which itself is used to accomplish a primary task. In general, its **audience** is all users and their **goal** is to understand the tool itself.

In focussing upon secondary sources, we first note that the user's goal—understanding the primary application—takes different forms depending on his or her situation. Thus we can further classify secondary sources as follows:

---

[1] An application interface is itself a species of online communication, including its menu design, the system and error messages, and its other user dialogs. In addition, there are intelligent interfaces, which attempt to integrate help into the interface rather than provide access to information. Our emphasis upon user-accessed information, therefore, is an important one: an extended treatment of online communication in general is beyond the scope of this essay (but see, for example, Brueker & De Greef, 1985).

Three types of secondary sources (electronic documentation)

1. I want to *buy* it. The exemplar for information that meets this goal is the sales demonstration. The **audience** is usually prospective buyers. Their **goal** is to (perhaps) buy the product or to understand how they can use its services; ultimately, the information reflects a persuasive aim.

2. I want to *learn* it. The exemplars for this category include tutorials and guided tours (which may emphasize a successful first experience—the affective response of the user—as an adjunct to the goal of learning). The users' **goals** are to learn what they can do with an application and how they perform some set of important or fundamental tasks. Constraints on attention generally limit this set of tasks to a group of basic skills, although more elaborate forms of instruction, much like a course or curriculum, also exist. From the user's point of view, however, the distinguishing feature is the goal of learning rather than performing. The **audience** includes both the prospective buyer and the novice or transfer user. Typically, the **context** is a set of artificial situations constructed so that the user can practice using the application; the user is not actually performing his or her own work.

3. I want to *use* it. The exemplars for this category include online help or online documentation, although neither of these labels identifies the necessary functionality required of the information. Critically, the **context** is the actual work situation where the user is trying to perform a task with the application. The users' **goals** are to overcome impasses that prevent them from proceeding on their task. The **audience** is all users, depending on the type of knowledge they require. For example, the novice typically needs task-oriented information, while the expert wants access to reference material.

For our purposes, the difference between a learning-oriented aim and a performance-oriented aim is critical. We restrict our use of the term online help to systems that support performance; online tutorials and training support the goal of learning. It is important to understand that implementations can certainly mix the two functions in one system—designers might allow access to a tutorial from help, or deliver both via some other framework. Ours is a functional definition, since we believe that the reasons users come to help should determine the way in which help is delivered.

Characteristics of performance-oriented support systems

In studying performance-oriented systems (like online help), we really want to understand

1. how the user accomplishes tasks with the application,

2. how the user deals with breakdowns in task performance, and

3. how the help system can get the user back on task.

Note that the user will not try to access performance-oriented information as long as the task is going smoothly. All users, however, encounter breakdowns (Winograd & Flores, 1986); that is, they reach points in tasks where they lack the procedural or semantic knowledge to continue. An analogous situation is our use of a car. Under normal operation, we do not interact with the car as a tool, but use it to take us places. But when the car breaks down, we must interact with it as a tool: Is the battery dead? Must I change a tire? For some breakdowns, the only recourse is to take the car to a specialist who can both diagnose the problem and then repair it.

A similar series of events occurs when the goal-directed activity of the user breaks down. After Brown & VanLehn (1987), we call such a breakdown an *impasse*. On encountering an impasse, users can no longer interact with the application as a tool for accomplishing a task; rather, the impasse itself becomes the focus of attention. To continue, users must *diagnose* their problem and then access knowledge that will allow them to *repair* the impasse. That is, the users must engage in secondary problem solving. If they lack the requisite knowledge directly (that is, in long term memory or via inferencing), users may turn to secondary sources for the information (they can also, for example, ask other people or experiment with the system).

Performance-oriented information, then, has two primary functions:

- to aid diagnosis by allowing access to information based on the *types* of impasse a user will encounter; and

- to aid performance by providing specific information that gives the user one (or more) repair strategies.

Traditionally, online help systems provide little support for the diagnosis stage[3]. For example, a user may represent his impasse as a procedural question (*How can I get a listing of my documents?*), but only access an alphabetical list of topics, a list that, in the worst case, contains just the

---

3. Clearly error messages, which are not under the users' direct control, are an integral part of the diagnosis stage. That error messages often do not help users indicates that professional writers desperately need to claim error messages as their domain.

commands from the application program. He must then decide which of the various commands will best answer his question.

As designers, we need to change our strategy: first, we must determine the information that users need to know, then we must present that knowledge in ways that meet their expectations. But how does one characterize the kinds of information needed by the user? Of the many ways to view human-computer interaction—internal-external task mapping (Moran, 1983), Cognitive Complexity Theory (Kieras & Polson, 1985), Task-Action Grammars (Payne & Greene, 1986), and GOMS (Card, Moran, & Newell, 1983)—we have found Ben Shneiderman's model of user knowledge a useful way to represent the information needs of the user (see Figure 1).

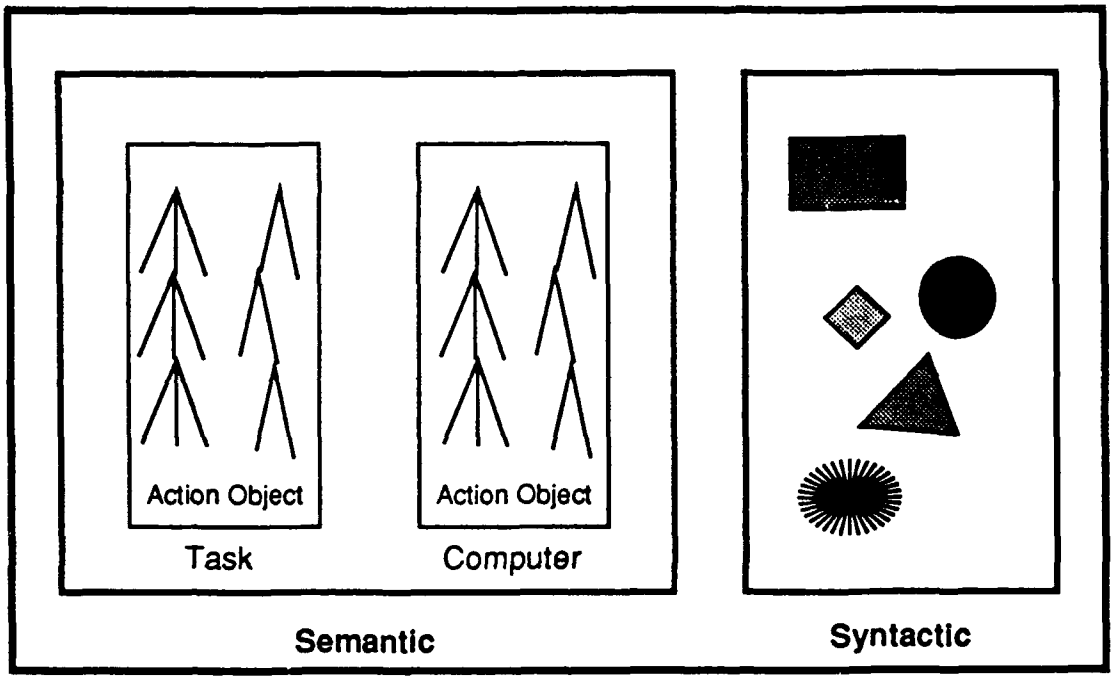


Figure 1: A representation of user knowledge in long-term memory (based on Shneiderman, 1987, p. 43).

Shneiderman's model allows for a syntactic and semantic component to a user's knowledge. The syntactic component includes all the device-specific knowledge necessary to perform actions. The semantic component involves

- real-world tasks and actions;
- computer tasks and actions (for example, using two windows to display different parts of a document) and
- a mapping between the two.

For example, a real-world writing task might be to *compare* (action) two parts of the same *document* (object), perhaps in order to check the citations against the reference list. Similarly, there might exist a computer task that allows users to *split* (action) a *window* (object) into two parts or *subwindows* (object). Related to this action is the appropriate syntax for performing it (in the current environment)—perhaps by typing the name of a command called "split window," by pressing 2 while holding the control key (^2), or, with a direct manipulation interface, by dragging an icon to the middle of the window. Finally, users must learn to map the real-world task into the corresponding computer task or tasks: to compare two parts of the same document, one can split the current window, and so on.

But how do we relate information to these types of knowledge? The user's knowledge of specific actions, objects, and syntax matches descriptive information, for example, a command reference that details all the commands in a system. The Unix **man** system, for example, provides online information for all the possible Unix commands. Each **man** unit includes a description of the command name, its function, a synopsis of its syntax, and related commands. The focus is on documenting the commands, not on how to use them to achieve real-world tasks. The basic **man** system even fails to help users relate computer objects and actions to their syntax. If users know that they want to list the contents of their directory, but they do not know some syntax—the specific name of the command for doing this, **ls**—they cannot get help.

The user's knowledge of the tasks they want to accomplish matches task-oriented, or procedural, information. Again, most help systems provide procedural information that supports computer tasks, such as how to split a window or how to open a new file. Few systems, however, relate real-world tasks to the computer tasks. That is, a user may know that she wants to move various pieces of text to another document (a task represented using real-world actions and objects). But the help system may only support computer tasks such as splitting windows, opening files, and cutting and pasting text. Once the user discovers the correct computer semantics—for example, that splitting the window is part of the task—she still must correctly order the various subtasks to achieve the overall task.

Obviously the help system cannot anticipate every real-world task that users will want to accomplish. It is important, therefore, to determine a range of prototypical real-world tasks using task analysis. From these tasks (and concrete examples), users will be able to apply their knowledge to novel situations. A task-orientation, therefore, will help users to learn the necessary computer semantics and syntax.

Medium constraints on help information

We have defined online help based on the needs of users in specific situations. In this sense, our description is implementation-independent; certain aspects of it apply to either online or hardcopy information. We chose this approach to emphasize the functional role of information—what does the user need to know? We do not want to suggest, however, that putting information online does not present special problems over and above the specification of content. We recognize that the constraints of presentation online have great impact on the design and evaluation of content as implemented in any one form (cf. James, 1985; Brockmann, 1986).

Based on interviews we conducted with nine professional writers who had at least five years of experience in the computer industry and who had primary responsibility for developing at least one online help system, we can begin to make several assertions about the difference between the online and hardcopy mediums (Duffy et al., 1988). Constraints imposed by the new medium include

Screen size — Writers disagree on whether online help should be formatted in the same way as hardcopy documentation. Certainly, limited screen size forces writers to chunk information in new ways. In this respect, writers suggest that a computer screen demands physical structuring as well as conceptual structuring even more than the page does because of the disorientation that many users experience when working with an online document.

Access mechanisms — The traditional hardcopy Table of Contents and Index do not transfer easily to computer. Writers are often responsible, not only for the content of the help system, but for the organization of and access to that help information.

Memory limitations — Although technological advances in this area are diminishing the importance of this constraint, memory still figures into the design of online information. The constraints of the software and hardware influence the amount of information designers can provide online.

Lack of context — There is no parallel navigation problem in a hardcopy document because the physical manual provides a "compass" with which users can orient

themselves to other sections of the documentation (for example, "Now I am in the front of the book; now I am in the index"). Unlike the users of hardcopy, users of online information tend to "get lost in space."

Summary

We have defined online help based on the type of information the user needs, taking into account the various constraints presented by the new medium. Now we want to consider methods for evaluating existing online help systems . Then we will present our method of evaluation based on an analysis of user's tasks.

3. Goal and methods for evaluating online help systems

In this section we discuss the goals and methods for evaluating online help systems. In particular, we consider how the choice of goals determines which data are relevant for the evaluation. We then examine some of the constraints on the evaluation process and discuss alternative evaluation methods in light of these constraints.

The goals of evaluation

As defined in section 2, an online help system should support the secondary problem solving behavior of its users. Users come to a help system after reaching an impasse with the primary application. A *usable* help system, then, is one that aids the user in diagnosing and repairing their problem (impasse) with a minimum expenditure of time or effort.

When we evaluate help systems, our primary goal is to provide an operational definition of usability. That is, we want to provide a set of specific criteria which, taken together, indicate usability. Our secondary goal is to develop an instrument that is robust—one, in short, that meets the needs of both the researcher and the practitioner. We recognize that practitioners typically do not have the resources to engage in large-scale experiments or studies.

But what factors make a help system usable? For example, it is not sufficient to determine whether users can find the necessary information; the evaluation must determine how *efficiently* users can find this information. Moreover, as Borenstein (1985) stresses. one must certainly consider the quality of the information itself. Is it well written? Is it accurate? We argue that questions regarding usability fall into four basic categories:

1. the accuracy of the information presented;

2. the completeness of the information presented;

3. the ease of accessing the information; and,

4. the ease of understanding of the information.

A significant deficit in one category tends to make the system unusable. Providing a complete database of information matters very little if that information is inaccurate, and neither completeness nor accuracy matter much if the text is so poorly written that the user cannot understand it. Even if the information is easy to understand, it will not help if the user cannot find it, as when tasks are not represented explicitly or when relevant information is scattered throughout the system.

Our goal for evaluation, then, is to rate help systems based on the criteria of accuracy, completeness, access, and comprehensibility. Given this goal, we must now develop a methodology for achieving it. The appropriate methodology, however, partly depends upon the audience for the evaluation, that is, who will use the data.

The audience constraint

We can define two specific audiences for our evaluation data. The first audience includes the developers or designers of the help system. They may use many types of evaluation to provide feedback about their design. In general, this type of evaluation is called *formative evaluation*. Formative evaluation aims to guide the iterative process of design and revision: it may unearth problems with a design and, potentially, identify strategies for solving those problems.

The second audience for evaluation includes users and their representatives (those who review the product in order to advise customers on its advantages or disadvantages). The customer who wants to buy an application program should have data about the usability of its help system. Similarly, software reviewers for magazines should not only review the application program, but also the strengths and weaknesses of its help system. In general, evaluation of an existing product is called *summative evaluation*. Summative evaluation happens independently of the developers. Since the primary goal of summative evaluation is to rate the usefulness of an existing product, it should allow the reviewer to compare help systems against one another or against some reference standard.

Methods of evaluation

We can now examine evaluation methods in light of the above goals and audience constraints. Schriver (1988) identifies three groups of methods used to evaluate documentation:

1. expert review
2. document analysis, and
3. performance assessment.

Expert review includes all the ways in which both subject matter experts and design experts review a document. Document analysis includes such strategies as the use of readability formulas, syntactic analysis, and discourse analysis. Finally, performance assessment involves examining the performance of intended users of the system as they work with it. Most of these evaluation strategies are methods for formative evaluation.

Protocol-Aided Revision. Schriver (1988) argues that one variety of performance evaluation, protocol-aided revision (or PAR), is more effective than either expert review or document analysis for formative evaluation. In the PAR approach to the evaluation of online help, one first identifies tasks for users to perform with the application. The end users are then asked to complete those tasks using the help system as their only resource. Importantly, the end users are instructed to think aloud as they work on the task. That is, they are asked to verbalize their thoughts. From a transcript of the session, the evaluator tracks how users formulate their need for help (when they reach an impasse in the application program), how they search for information, and how they apply that information to overcome their impasse.

While we agree with the value of PAR for formative evaluation, the approach does not easily provide comparative data for summative evaluation. Think-aloud protocols yield data about the usability of a particular system; because of the verbalization, protocol analysis cannot be reliably used to compare either versions of the same system or different systems without greatly complicating the design of the test. For example, the evaluator must establish a valid and reliable system for categorizing comments into categories relevant to the comparative evaluation. Then he or she must develop a scoring system for each category, one that is not influenced by the "talkativeness" of a user. Finally, the evaluator must identify tasks that generalize from one system to another.

Our interest is in summative, rather than formative, evaluation. That is, our goal is to develop a strategy that will provide comparative data on the

effectiveness of help systems. We want a tool for reviewers and potential users of a system that allows them to (1) make detailed ratings of a help system; and (2) reference their ratings to a database of ratings of other help systems.

Benchmarking. Benchmarking is a form of summative evaluation that does aim to provide comparative data (Lewis & Crews, 1985). Like PAR, the evaluators first define a set of problems that can be solved on all the systems being tested. Then, they establish a specific experimental methodology to examine users performing those tasks. Finally, they collect speed and accuracy data for novice users. These dependent measures become the basic data for comparing different systems.

Roberts & Moran (1983) describe a benchmark for comparing word processors. They define a matrix of word processing *actions* (for example, delete, move, and copy) and *objects* (for example, characters, words, sentences, and paragraphs). From this matrix, they derive a set of basic word processing tasks and, using standard material and testing procedures, compare the accuracy and speed with which novice users do those tasks on different word processors. While the approach has received some criticism (Borenstein, 1985), others have used the database established in Roberts & Moran's (1983) original study to evaluate new word processors.

Unfortunately, developing a benchmark to evaluate online help is problematic. A benchmark uses a set of common tasks and compares the time and accuracy of performance. Initially, one can question whether it is possible to define tasks common to even a representative subset of help systems. After all, help systems document different application programs and therefore support different types of tasks. (As discussed later, however, we think that a cognitive task analysis of the user's encounter with a help system can form the basis for comparative evaluation.)

The main difficulty with benchmarking arises not with defining common tasks, but with its emphasis on speed and accuracy of performance. We do not believe that comparing the speed and accuracy of users on a task in the application will necessarily reflect on the usability of the help system. For example, a complex application program, that is, one with a large command set, will require a more complicated help system In this case we would expect slower and potentially less accurate performance, not because the help system is poorly designed, but because it is large and complex. Moreover, the nature of the application program (for example, a word processor versus an operating system) and the quality of the applications interface may also impact the help system negatively if the only measure of usability is speed and accuracy on a task.

4. A conceptual model for the evaluation of online help systems

In contrast to PAR and benchmarking, our approach to evaluation has two distinguishing features. First, our evaluation focuses on the tasks a user should be able to perform within the help system itself. Second, we distinguish between the evaluation of accuracy and completeness and the evaluation of access and comprehensibility. The accuracy and completeness of the help information are intimately related to the application software; one can only determine whether help accurately supports a complete range of tasks by fully testing its information against a list of those tasks.

One can determine, however, the ease of accessing and comprehending information in the help system independently of the application program. The basic concept is that the user of the help system always engages in certain tasks specific to using the help system, such as scanning a menu for topics. Even though the content of the help system should support the tasks of the primary application (word processing, database management, and so on), the actual use of the help system itself has many regularities.

A task analysis of the use of online help

What is the situation of the user when he or she turns to online help? As established in section 2, users are trying to complete some primary task using an application. When they reach an impasse, they turn to help in order to obtain information that will let them overcome the impasse. Task analysis identifies those tasks the user must perform to solve this secondary problem. Eight major tasks, as shown in Figure 2, are identified in the task analysis. The tasks are described as follows.

1. *Represent the problem.* Users will somehow label their impasse or information need. Although the required information is unknown, users usually won't stop with a statement of "I don't know what to do next." Rather they develop two hypotheses. First, they guess a likely name for the needed information. This naming process may be at a very general level, such as the class of activities involved, or it may be very specific, such as a likely command name. Second, users must guess how the help system will represent the needed information. That is, they will attempt to reformulate their representation of the impasse into terms that they expect to work in an information search task.

In using a help system, users must:

1. Represent the problem
2. Access the help system
3. Select a help topic
4. Scan the information
5. Understand the information presented
6. Select the needed information on the topic
7. Navigate to new or related topics
8. Transfer the information to the application

Figure 2: Primary tasks that users must perform while using a help system.

2. *Access the help system.* Given a representation of the problem, users must access the help system. Here we simply mean actually getting into the help system—a task that is a significant problem for some systems.

3. *Select a help topic.* Once in the help system, users must select a help topic that they feel will probably provide the information they need. In essence, they need to find a topic that explicitly matches their original representation of the problem or one that seems to provide the best fit given the impasse.

4. *Scan the information.* Once users get to a help topic that they selected, it must be scanned to determine if it contains the needed information. If the help information is just a text file with little discourse structure, scanning may be difficult if not impossible.

5. *Understand the information presented.* Once they locate a relevant section of information to read, users must be able to understand the text and graphics.

6. *Select the needed information on the topic.* If the information is on the correct topic and is understood, users must extract the particular information needed to complete their task in the application. This includes being able to

translate the general help information provided into the specific actions needed for the application.

7. *Navigate to new and related topics.* If they did not find all the needed information, users must go to another topic. In general, users must be able to navigate through the help system. Of course, as they come to new topics, users will step through tasks four, five, and six again. If they ultimately fail to find the necessary information, users may start over by reformulating their problem (or give up in exasperation).

8. *Transfer the information to the application.* Finally, users must take the information from the help system and apply it in the application program in order to remove the impasse.

When we speak of the usability of a help system we mean how well the help system supports each of these tasks. While our method for evaluating access and comprehensibility may differ from our method for evaluating access and completeness, both forms of evaluation must be framed in terms of this task analysis.

Evaluating usability: access and comprehensibility

The goal of summative evaluation is to provide comparative data on the usability of help systems. If we are to use the task analysis described above, we must be able to score any help system in terms of each of the eight user tasks. But what are we to score and how do we obtain reliable and comparable scores?

First, let us consider what we are to score. If we focus on the access and comprehensibility components of usability, then we must score each user task in terms of how easy it is to access and understand the information that is relevant to performing that task. For example, in terms of the first task, Problem Formulation, we may assess how well the system tries to match the user's formulation of the problem.

We could assess the system through user testing. However, as we have argued earlier, the user's speed and accuracy of performance in the help system is significantly affected by the complexity, generality, and design of the application program. Instead, our evaluation system employs expert judgement to provide the basis for the evaluation of access and comprehensibility. Of course, if we are to have comparable ratings, then the expert judgement must be tightly constrained. In essence, the expert must have very specific criteria on which to rate the system.

We are using specific guidelines for effective design as the mechanism for constraining or guiding the expert judgment. We have taken the guidelines for effective design from the literature on help systems, interface design, and document design (for example, Smith & Mosier , 1986; Kearsley, 1988; Meyhew, in press; Duffy, Palmer, & Mehlenbacher, in preparation) and recast them as statements to be used in rating a help system. While there is some question of the guideline approach as a means of directing the activities of a writer or designer (Duffy & Kabance, 1982; Duffy, 1985b; Mosier & Smith, 1986), guidelines have been used to evaluate products successfully (Duffy, 1985a). That is, well designed material will exhibit characteristics recommended by guidelines.

We use two criteria for selecting guidelines. The first criterion is that the guideline should indicate reduced information processing demands on the user. A usable help system is one in which the user can obtain information with a minimum of time and effort, that is, with minimal information processing. Therefore, the guidelines used to judge a help system should clearly reflect a reduction in information processing requirements. This may happen by reducing the demands on any one of the three human information processing subsystems: perceptual, motor, or cognitive.

The second criterion is that the guideline should address one (or more) of the eight tasks identified in our task analysis. For example, guidelines having to do with the organization of items on a menu would be grouped under user Task Three, *"Select a Topic,"* while guidelines addressing the nature of the items on the menu (or in a keyword system) would be classified under Task One, *"Represent the problem."*

The final evaluation system consists of 49 items arranged into categories based on their relevance to the eight user tasks. The items guide the expert to review the help system thoroughly and to rate it based on the presence or absence of particular features (for example, help on help) or on the judged quality of a particular feature (for example, the consistency of format across help topics as one index of Task four—the ease of searching a topic to find the needed information).

The system yields scores that can range between 0 and 100 for each user task as well as for the system overall. Currently, the evaluation system is in the final stages of development. Two reviewers are evaluating 25 help systems for commercially available software. This data will provide reliability information as well as a database of scores that can be used in comparing other help systems. The validity of the system is difficult to assess empirically. We have already mentioned the limitations of user performance data; it cannot, therefore, serve as a criterion for the validity of the rating system. We argue that the validity of our system depends on

- the degree to which the evaluation is based on a conceptually sound task analysis of "usability" and

- the degree to which the particular items comprise the concept of usability and sample the full range of usability factors.

Assessing usability: accuracy and completeness.

Access and comprehensibility are features of the help system that can be evaluated independently of the application software. In addressing accuracy and completeness, we must focus on the link between the help system and the application being helped. This discussion reflects our initial thinking on the issues involved in assessing accuracy and completeness.

Let us begin by relating the issues of accuracy and completeness to our analysis of the tasks of the user entering a help system. We see three of those tasks as being relevant to the evaluation of accuracy and completeness.

Represent the Problem	Are the tasks the user needs help on represented in the help system? The issue for evaluating access and comprehensibility is whether the topics are framed in the user's terms. Here we are concerned with the number of tasks covered. How comprehensively does the help system address the use of the application program? Is the user's goal represented at all?
Select Needed Information	Is the information on a task accurate and complete? Here we are concerned with the completeness of the coverage of any one topic or task. A complete coverage of the procedure for executing a task would list each action the user should take and describe the consequence of each of those actions. The clarity of the description and the availability of graphics which help the user to locate key information are all issues of comprehensibility; that the steps are present is the requirement for completeness. Complete procedural coverage is usually provided only for novice tasks or startup tasks (for example, booting the system). For some reason, it is assumed that users do not need this procedural support for more advanced functions—an assumption we take strong exception to. Complete coverage of a topic (depth) also means complete description of functions and complete coverage of all "relevant" technical data.
Navigate to	Is information relevant to the user's goal identified

New Topics

with that goal and co-located or clearly linked? It does little good if the relevant information is in the help system but the user will never find it because it is linked to a different task (from the point of view of the user) or the bits of necessary information are in different places with no cross referencing. In evaluating comprehensibility and access we focus on general navigation aids—aids to permit easy navigation and prevent getting lost. Here the concern is the linkage between specific chunks of information required to complete a task in the application.

Before considering methodologies for evaluating each of these issues, let us consider some of the problems we face in the evaluation of accuracy and completeness. We have defined the breadth of tasks covered—the issue of problem representation—in terms of the users' tasks. The first issue to address is whether or not this is the correct unit of analysis. Should everything be cast in terms of user tasks? We think so. But while we can argue for task orientation in defining a user's needs, we face a second problem, that of defining a task. At what level should a task be conceptualized? In part this is an issue of grain size—is the task cutting, cutting and pasting, or revising. In part it is also a matter of domain representation—are we talking about computing tasks or real-world tasks (Shneiderman, 1987)? There are no ready answers to this issue. However, movement towards an answer must be based on an analysis of how users conceptualize their task or goal in the application.

A third issue involves how well the help system covers the breadth of tasks which the application supports. Even if we agree to represent tasks in the help system, and even if the tasks are defined in terms of grain size and domain, we still must recognize that there are an infinite number of tasks. That is, the application will always be used in new and unanticipated ways by the end users and thus we can never specify all of the possible user tasks. How are we to evaluate the adequacy of the help system for unanticipated tasks?

In part the evaluation can address the wide variety of possible tasks by sampling rather than exhaustively enumerating tasks. That is, we must sample a range of potential tasks in the application and then determine the proportion of those tasks addressed in the help system. Of course the sampling strategy will be critical here. However, this still does not help us evaluate the adequacy of the help system for aiding the user in new, innovative tasks.

Online help systems address the problem of unanticipated tasks by providing reference information on each command in the application. The evaluation

of this reference information must be in terms of how well it supports the users needs. This requires identifying the kinds of reference information that must be provided. Of course the designers should also have addressed this issue, but too often this is not done in a systematic fashion. Indeed, the very fact that reference information is organized around commands rather than around the categories of knowledge the user requires in these innovative tasks is an indication of the lack of systematic attention to this issue. What is required is a theoretical and empirical analysis of the categories of knowledge required of the user. This analysis would then serve as the basis for evaluating completeness.

We have been focusing on completeness in terms of the breadth of coverage in the help system. However, there are also methodological issues in evaluating accuracy and completeness of the content provided on any one topic. Given a task in the application program, we now need a method for evaluating the accuracy and completeness of the information on that task. A straight forward procedure would be to simply test the ability (speed and accuracy) of users to perform a task in the application with only the help system available as a source of information. However, there are two problems with this approach. First, the speed and accuracy data will reflect the effects of the comprehensibility and accessibility of the information as well as the accuracy and completeness. Indeed, a user may take a long time or never complete the task simply because he or she cannot locate (access) the relevant information.

One might argue that this approach, while not distinguishing between the components of usability, does, in fact, index the overall usability of the help system and that this, after all, is our goal. However, our goal is not just to assess usability but to assess it in such a way that we can compare help systems. This leads us to the second problem with the use of speed and accuracy in using the help system: these measures will be dependent on the complexity of the application program. The more complex the application program, the longer it will take to perform a task and the more likely that there will be an error in that performance. Thus the overall indices of speed and accuracy cannot be compared across help systems.

We do not think that the evaluation needs to be, or even should be, performance based. Rather, for each task we abstract all relevant information from the help system. We then want to assess the completeness and accuracy of the procedural and conceptual information. Procedural accuracy and completeness can be assessed by comparing the steps required to perform a task with the steps provided in the help system: what proportion of the steps are presented? Conceptual accuracy and completeness can be assessed by identifying the concepts required to perform the procedure and checking the proportion that are "explained" in the help system.

Finally, the assessment of the information and of the navigation proceeds by identifying the proportion of information required for a task that is linked either through co-location or through explicit cross-referencing. Note, however, that the analysis of navigation requires the specification of a user's goal as the primary mechanism for defining a domain of information. Clearly, we recognize that the development of a system for evaluating accuracy and completeness holds more difficult issues—both conceptual and practical—than the evaluation of access and comprehensibility. However, we feel that the focus on the user in terms of user tasks and user knowledge requirements is the key to developing an effective evaluation tool. Also, we should note that the evaluation instrument will not work in isolation; its success will depend on the availability of expert users of the application, on detailed assessment of tasks, and on the use of the help information in the application program.

4. Conclusions

Our goal in this paper has been to review some of the problems posed by online training and assistance and to provide a conceptual model for understanding online help systems. In describing our framework, we have categorized various types of electronic documentation and restricted our definition of online help to systems that are performance-oriented rather than learning-oriented. Performance-oriented information, we have suggested, has two primary functions: to provide information based on the types of impasses users encounter and to provide users with one (or more) repair strategies. We have not denied the constraints presented by the new medium—that is, considerations of screen size, access mechanisms, memory limitations, and lack of context—but rather, we have stressed that online help systems should be viewed as a mixture of content (knowledge required by the user) and constraints imposed by online presentation.

Because we believe that, when talking about online help systems, the focus should be on the user's tasks, we similarly maintain that evaluation should emphasize how effectively the system reduces the user's motor, perceptual, and cognitive demands. With this task-oriented perspective in mind, we have surveyed various evaluation techniques—for example, PAR and benchmarking—and found that a more comprehensive methodology is required. That is, existing methods of evaluation do not capture the elements of a help system that we have suggested make it "usable." A comprehensive evaluation of online help must account for the system's effectiveness in providing users with information that is accurate, complete, easy to access and easy to understand.

Finally, we believe that our model, which is based on an analysis of users' cognitive tasks while accessing an online help system, gives both researchers

and practitioners a theoretical understanding that can potentially be used to guide both the design and the evaluation of online help systems.

References

Bethke, F., Dean, P., Kaiser, E., Ort, E. & Pessin, F. (1981). Improving the usability of programming publications. *IBM Systems Journal*, 20 (3), 306-320.

Borenstein, N. (1985a). *The design and evaluation of on-line help systems.* Unpublished Doctoral Dissertation. Pittsburgh, PA: Carnegie Mellon University.

Borenstein, N. (1985b). The evaluation of text editors: A critical review of the Roberts and Moran methodology based on new experiments. In *Computer-human interaction (CHI) '85 proceedings.* New York, NY: Association for Computing Machinery.

Brockmann, R. J. (1986). *Writing Better Computer User Documentation: From Paper to Online.* NY: John Wiley & Sons.

Brown, J. S. & VanLehn, K. (1980). Repair theory: A generative theory of bugs in procedural skills. *Cognitive Science, 4,* 379-415.

Breuker, J. & de Greef, P. (1985). *Information Processing Systems and Teaching & Coaching in HELP Systems.* Deliverable 12.1, ESPRIT Project P280 (EUROHELP), Amsterdam: Department of Social Science Informatics, University of Amsterdam.

Card, S. K., Moran, T. P., & Newell, A. (1983). *The psychology of human-computer interaction* Hillsdale, NJ: Lawrence Erlbaum Associates.

Carroll, J. M. & Mazur, S. A. (1985). *Lisa learning.* (Tech Report RC 11427). Yorktown, NY: T. J. Watson Research Center.

Duffy, T.M. (1985a). Readability formulas: What is the use? In T.M. Duffy and R. Waller (Eds.), *Designing Usable Texts.* New York: Academic Press.

Duffy, T.M. (1985b). Controlling the usability of technical text: Specifications and guidelines. In D.H. Jonassen (Ed.), *The Technology of Text. Vol. 2.*

Duffy, T.M. & Kabance, P. (1982). Testing a readable writing approach to text revision. *Journal of Educational Psychology, 74,* 733-748.

Duffy, T., Palmer, J., & Mehlenbacher, B. (in preparation). *Online help systems: Design and evaluation.*

Duffy, T., Gomoll, T., Gomoll, K., Palmer, J., & Aaron, A. (1988). *Writing online information: Expert strategies (interviews with professional writers).* Pittsburgh, PA: Communications Design Center.

Elm, W. & Woods, D. (1985, October). Getting lost: a case study in interface design. *Proceedings of Human Factors Society.* Westinghouse Scientific Paper: 85-1C60-CONRM-P2.

Haas, C. & Hayes, J. R. (1987). *Effects of Text Display Variables on Reading Tasks: Computer Screen vs. Hard Copy.* Technical Report CDC-3 (ERIC # 260 387), Pittsburgh, PA, Communications Design Center, Carnegie Mellon.

James, G.. (1985). *Document Databases: The New Publications Methodology.* New York, NY: Van Nostrand Reinhold.

Kearsley, G. (1988). *Online help: Design and implementation.* Menlo Park, CA: Addison-Wesley Publishing Co.

Kieras, D. E. & Polson, P. G. (1985). An approach to the formal analysis of user complexity. *International Journal of Man-Machine Studies, 22,* 365-394.

Kinneavy, J. L. (1970). *A theory of discourse.* Englewood Cliffs, NJ: Prentice-Hall.

Kruk, R. & Muter, P. (1984). Reading of continuous text on video screens. *Human Factors,* 26 (3), 339-345.

Lewis, B. & Crews, A. (1985, March). The Evolution of Benchmarking as a Computer Performance Evaluation Technique. MIS Quarterly, 7-16.

Marshall, C., Nelson, C., & Gardiner, M. (1987). Design guidelines. In M. M. Gardiner and B. Christie (Eds.), *Applying cognitive psychology to user-interface design.* New York, NY: John Wiley and Sons.

Meyhew, D. (in preparation). *Principles and methods of interactive system design.*

Moran, T. P. (1983). Getting into a system: External-internal task mapping analysis.. A. Janda (Ed.), *Proceedings of the CHI '83 Conference on Human Factors in Computing Systems* (pp. 45-49). New York: ACM.

Mosier, J. and Smith, S. (1986). Application of guidelines for designing user interface software. *Behaviour and Information Technology, 5,* 39-46.

Muter, P., Latremouille, S.A., Treurniet, W.C., & Beam, P. (1982). Extended reading of continuous text on television screens. *Human Factors*, 24, 401-414.

Ong, W. (1982). *Orality and literacy: The technologizing of the word.* London: Methuen.

Payne, S. & Greene, T. (1986). Task-action grammars: A model of the mental representation of task languages. *Human-Computer Interaction*, 2, 93-133.

Roberts, T. and Moran, T. P. (1983). The evaluation of text editors: Methodology and empirical results. *Communications of the ACM*, 26, 265-283.

Robertson, C.K. and Akscyn, R. (1982). *Experimental evaluation of tools for teaching the ZOG frame editor.* Technical Report. Pittsburgh, PA: Carnegie Mellon University.

Schriver, K. (1988). *Teaching writers to anticipate the readers needs: Empirically based instruction.* Unpublished PhD dissertation. Pittsburgh, PA: Carnegie Mellon University.

Shirk, H. N. (1988). Technical writers as computer scientists: The challenges of online help. In E. Barrett (Ed.), *Text, ConText, and HyperText: Writing with and for the computer* (pp. 311-27). Cambridge, MA: MIT Press.

Shneiderman, B. (1987). *Designing the user interface: Strategies for effective human-computer interaction.* Reading, MA: Addison-Wesley.

Smith, S. and Mosier, J. (1986). *Guidelines for designing user interface software.* (ESD-Technical Report TR-86-278). Hanscom Air Force Base, MA: Electronic Systems Division, United States Air Force.

Walker, J. (1987). Issues and strategies for online documentation. *IEEE Transactions on Professional Communication, PC 30*, 235-248.

Winograd, T. & Flores, F. (1986). *Understanding computers and cognition: A new foundation for design.* Norwood, NJ: Ablex.